

**UNIVERSIDAD COMPLUTENSE DE MADRID**

**FACULTAD DE INFORMÁTICA**



**PROYECTO SISTEMAS INFORMÁTICOS**

**CONTROL INTELIGENTE DEL**

**PÉNDULO INVERTIDO**

**2011-2012**

**Autores:**

**Alberto Hernández Largacha**

**Marco Legaspi Martínez**

**Jaime Peláez Martín**

**Directores:**

**Matilde Santos Peñas**

**José Antonio Martín Hernández**



## **Autorización**

Los alumnos, Alberto Hernández Largacha, Marco Legaspi Martínez y Jaime Peláez Martín, matriculados en la Facultad de Informática, autorizan a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria como el código, la documentación y/o prototipo del presente trabajo de Sistemas Informáticos: "CONTROL INTELIGENTETE DEL PÉNDULO INVERTIDO", realizado durante el curso académico 2011-2012 bajo la dirección de Matilde Santos Peña y José Antonio Martín Hernández en el Departamento de Arquitectura de Computadores y Automática, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Madrid 25 de Junio de 2012

Fdo. Alberto Hernández Largacha

Fdo. Marco Legaspi Martínez

Fdo. Jaime Peláez Martín





## **Resumen**

Presentamos el diseño e implementación de un robot basado en la plataforma LEGO Mindstorm NXT. El sistema es capaz de mantener el equilibrio y avanzar sobre sus dos ruedas gracias a la información que recibe de sus sensores. Se elabora un modelo matemático del prototipo construido utilizando el método de Lagrange, se simula, y se diseña y sintoniza una ley de control para el modelo propuesto combinando diferentes estrategias de control como el control PID, LQR y la lógica borrosa. Para la implementación se utiliza la herramienta de desarrollo de aplicaciones RobotC.

## **Abstract**

We present the design and implementation of a robot based on LEGO Mindstorms NXT platform. The system is capable of maintaining balance and moving on two wheels based on the information it receives from its sensors. It develops a mathematical model of the prototype built using the Lagrange method, which has been simulated, and designed a control law for the proposed model based on different control strategies as PID control, LQR and fuzzy logic. To implement the tool is used RobotC application development.

## **Palabras clave**

Segway, Control péndulo invertido, Lego NXT, Lógica Fuzzy, Controlador PID, Regulador LQR, Modelado y Simulación.

## **Keywords**

Segway, Controlling inverted pendulum, Lego NXT, Fuzzy Logic, PID Controller, LQR Controller, Modeling and Simulation.



# Índice general

## Índice Figuras

## Índice Tablas

### 1 Introducción

- 1.1 Descripción del problema
- 1.2 Objetivos
- 1.3 Estructura de la memoria

### 2 Estado del arte

- 2.1 Control del péndulo invertido
- 2.2 Control PID
- 2.3 Control ganancia programada
- 2.4 Regulador LQR
- 2.5 Lógica Fuzzy

### 3 Construcción del péndulo invertido: Lego NXT

- 3.1 Otras configuraciones del Lego

### 4 Modelado matemático del sistema

- 4.1 Obtención del modelo matemático del péndulo invertido
  - 4.1.1 Parámetros del sistema
  - 4.1.2 Base matemática del sistema
  - 4.1.3 Modelo matemático en el espacio de estados
- 4.2 Simulación del modelo

### 5 Diseño del control del péndulo

- 5.1 Regulador LQR
- 5.2 Controlador PID de ganancia programada
- 5.3 Control basado en Lógica Fuzzy

### 6 Simulación e Implementación

- 6.1 Simulación de los controladores en Matlab
  - 6.1.1 Simulación del sistema en lazo abierto
  - 6.1.2 Control LQR
  - 6.1.3 Control LQR y PID
  - 6.1.4 Control PID de ganancia programada Fuzzy
    - 6.1.4.1 Zonas de trabajo para el controlador PID
    - 6.1.4.2 Módulo Fuzzy
- 6.2 Implementación del control en el sistema real

### 7 Conclusiones y Trabajos futuros

### 8 Referencias bibliográficas



# Índice figuras

## 1 Introducción

Figura 1.1 Esquema péndulo invertido

Figura 2.1 Ejemplo Segway

## 2 Estado del arte

Figura 2.1 Péndulo de Furuta

Figura 2.2 Joe

Figura 2.3 Balibot

Figura 2.4 nBot

Figura 2.5 Legway

Figura 2.6 Watanabe

Figura 2.7 Esquema controlador PID

Figura 2.8 Esquema realimentación

Figura 2.9 Esquema regulador LQR

Figura 2.10 Funcionamiento de un sistema

## 3 Construcción del péndulo invertido: Lego NXT

Figura 3.1 Giróscopo NXT Gyro de Hitec

Figura 3.2 Robot LEGO NXT en configuración de péndulo invertido

Figura 3.3 Ejemplo configuración I

Figura 3.4 Ejemplo configuración II

## 4 Modelado matemático del sistema

Figura 4.1 Esquema del modelo matemático

Figura 4.2 Fotografía del robot

Figura 4.3 Esquema del modelo matemático en espacio de estados

Figura 4.4 Diagrama de bloques del sistema en lazo abierto

Figura 4.5 Gráfica de inclinación del cuerpo del robot. Simulación en lazo abierto

## 5 Diseño del control del péndulo

### 5.1 Regulador LQR

### 5.2 Controlador PID de ganancia programada

Figura 5.2.1 Zona A

Figura 5.2.2 Zona B

Figura 5.2.3 Zona C

### 5.3 Control basado en Lógica Fuzzy

Figura 5.3.1 Entradas y salidas fuzzy

Figura 5.3.2 Entrada del ángulo del robot

Figura 5.3.3 Entrada velocidad angular

Figura 5.3.4 Velocidad de la rueda

Figura 5.3.5 Formas sistema borroso

Figura 5.3.6 Sistema de reglas

Figura 5.3.7 Superficie de control Zona A



Figura 5.3.8 Superficie de control Zona B

Figura 5.3.9 Superficie de control Zona C

## **6 Simulación e Implementación**

Figura 6.1 Diagrama de bloques del sistema en lazo cerrado

Figura 6.2 Gráfica evolución del ángulo de inclinación. Simulación lazo cerrado.

Figura 6.3 Diagrama de bloques del sistema con control LQR

Figura 6.4 Gráfica evolución del ángulo de inclinación. Simulación control LQR.

Figura 6.5 Gráfica evolución de la velocidad angular de las ruedas. Simulación control LQR.

Figura 6.6 Diagrama de bloque tras la inclusión del controlador PID

Figura 6.7 Gráfica evolución del ángulo de inclinación. Simulación control PID.

Figura 6.8 Gráfica comparativa de inclinación entre control LQR y LQR + PID

Figura 6.9 Gráfica evolución de la velocidad angular de las ruedas. Zona A

Figura 6.10 Tiempo de subida

Figura 6.11 Gráfica evolución de la velocidad angular de las ruedas. Zona B

Figura 6.12 Gráfica evolución de la velocidad angular de las ruedas. Zona C

Figura 6.13 Gráfica comparativa de las velocidades angulares aplicadas a las ruedas para las tres zonas.

Figura 6.14 Esquema de entradas y salidas del módulo fuzzy

Figura 6.15 Esquema de cálculo de la ganancia  $K_p$  del módulo PID

Figura 6.16 Esquema de cálculo de la ganancia  $K_d$  del módulo PID

Figura 6.17 Evolución de la ganancia  $K_p$  al pasar por diferentes zonas de trabajo

Figura 6.18 Evolución de la ganancia  $K_d$  al pasar por diferentes zonas de trabajo.

## **7 Conclusiones y Trabajos futuros**

Figura 7.1 Prototipo en la posición inicial

Figura 7.2 Prototipo avanzando

Figura 7.3 Prototipo subiendo una pendiente



## **Índice tablas**

Tabla 4.1 Parámetros constantes del sistema

Tabla 4.2 Parámetros variables del sistema

Tabla 6.1 Ganancias para cada área de trabajo

Tabla 6.2 Constantes

Tabla 6.3 Variables

# 1 Introducción

La temática de este proyecto se centra en el ámbito de la robótica, la automática y el control, seleccionando un sistema físico clásico como es el del péndulo invertido, el cual se trata de resolver con una combinación de diferentes estrategias de control ejecutadas sobre una plataforma real como es Lego Mindstorms NXT, del cual se dispone de un prototipo en el grupo ISCAR del departamento de ACYA.

El proyecto propone el uso no sólo de un control clásico sobre el sistema, sino una combinación de diferentes estrategias como el control PID de ganancia programada, el regulador LQR o la lógica fuzzy sintonizadas y coordinadas en base los datos obtenidos de la simulación del esquema matemático del sistema físico calculado previamente.

## 1.1 Descripción del problema

El péndulo invertido es conocido por ser uno de los problemas más importantes y clásicos de la teoría de control. El sistema se compone de un carro sobre el cual se monta un péndulo que puede girar libremente. El carro deberá moverse para compensar el desplazamiento del péndulo y mantenerlo, así, en equilibrio.

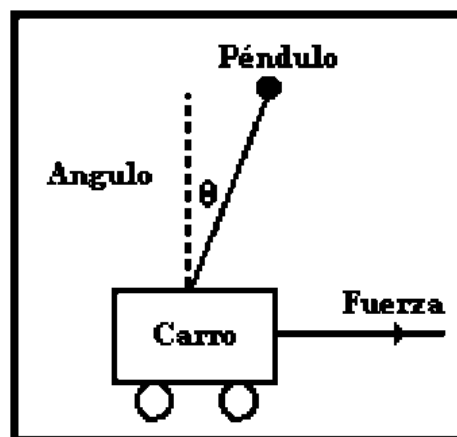


Figura 1.1 Esquema péndulo invertido

Su aplicación en la vida cotidiana se da desde en el control de estabilidad de grúas, hasta la construcción de vehículos de desplazamiento para humanos que implementan este problema, como es el caso de famoso vehículo *segway*<sup>1</sup>.



Figura 1.2. Ejemplo Segway

El problema aborda el control de un sistema inestable y altamente no lineal. A menudo es utilizado como ejemplo académico, principalmente por ser un sistema de control accesible, y por otro lado, permite mostrar las principales diferencias del sistema en lazo abierto y de su estabilización en bucle cerrado. Pese a existir diferentes técnicas a la hora de diseñar el regulador óptimo capaz de estabilizar el péndulo, no todas representan la mejor opción, aunque en muchas de ellas podamos encontrar ventajas que las demás no presentan.

El presente documento describe las instrucciones para la construcción de un robot en configuración de péndulo invertido, así como las estrategias que se pretenden seguir para el diseño e implementación de un controlador que combine diferentes técnicas de control clásicas. Lo más evidente para este modelo de planta es que los sistemas de control aplicables involucran cálculos y toma de decisiones rápidas para mantener los centros de masa de elementos del sistema en la posición y dirección correcta.

---

<sup>1</sup> *segway*: es un vehículo de transporte ligero giroscópico eléctrico de dos ruedas, con auto balanceo controlado por ordenador. El usuario se debe inclinar hacia la dirección que quiera tomar.



El tipo de control que se implementa para cada sistema depende de varios factores, entre ellos, si la planta a controlar es lineal o no lineal, estable o inestable, etc. Hasta hace pocos años el control de sistemas lineales se realizaba principalmente mediante reguladores Proporcional, Integral, Derivativo o una combinación de estos. Para el caso de sistemas no lineales, en especial de varias entradas y salidas, era común utilizar variables de estado. Hoy en día cada vez es más común utilizar controles “inteligentes” para realizar estas tareas.

## 1.2 Objetivos

El objetivo de este proyecto es la construcción de un robot en configuración de péndulo invertido, utilizando para ello la plataforma Lego Mindstorms NXT, el modelado matemático de este sistema físico, su simulación y el diseño, sintonía e implementación de un controlador para este robot que combine distintas estrategias de control para garantizar el equilibrio. Concretamente pretendemos implementar un PID de ganancia programada cuyas zonas de trabajo sean seleccionadas mediante lógica borrosa, el cual trabaja conjuntamente con un regulador LQR.

El control del robot debe garantizar el equilibrio en base a la información captada a través de su sensor de giro, y los encoders de sus dos motores, mediante la transmisión de potencia idónea a sus dos motores de corriente continua. Para ello, se realiza la construcción de un péndulo invertido sobre dos ruedas utilizando la plataforma Lego Mindstorms NXT, se elabora un modelo matemático del prototipo construido utilizando el método de Lagrange y se simula en el entorno Matlab<sup>2</sup>, y se diseña y sintoniza una ley de control para el modelo propuesto.

Para la implementación se programa la ley de control utilizando la herramienta de desarrollo de aplicaciones RobotC, con el objetivo de lograr el control del péndulo en la posición vertical.

---

<sup>2</sup> **Matlab:** software matemático que ofrece un entorno de desarrollo integrado con un lenguaje de programación propio (lenguaje M). Está disponible para las plataformas Unix, Windows y Mac OS X. El paquete Matlab dispone de dos herramientas adicionales GUIDE (editor de interfaces de usuario) y Simulink (plataforma de simulación multidominio).





### **1.3 Estructura de la memoria**

El proyecto se ha dividido en los siguientes capítulos: en primer lugar se hace una introducción de la temática del proyecto describiendo el problema, los objetivos funcionales de la aplicación y además especifica el diseño de la solución al problema propuesto.

A continuación se concreta la construcción del modelo, así como los sensores que se van a utilizar. Se define la estrategia de control basado en los modelos matemáticos.

En el siguiente capítulo se presentan los diferentes sistemas de control diseñado. Se presenta el péndulo invertido haciendo uso del control PID, regulador de pesos LQR, y lógica Fuzzy. Por medio de simulaciones se puede observar el comportamiento del controlador.

En el siguiente capítulo se muestra la implementación de la planta, y finalmente se presentan las conclusiones del proyecto.

## 2 Estado del arte

Los péndulos invertidos constituyen un banco de pruebas completo e interesante para la ingeniería de control. Uno de los más estudiados de esta familia de artefactos es el denominado péndulo invertido sobre un vehículo.

Desde los años 70 se han realizado varios proyectos con péndulos invertidos. Un investigador líder en esta área es el Profesor Furuta (Figura 2.1), quien desde entonces ha realizado notables aportes teóricos y experimentales concernientes a este problema de control [Furuta, 1976].

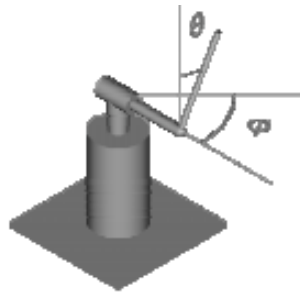


Figura 2.1 Péndulo de Furuta

Los investigadores Grasser et al. (2002) del Laboratorio de Electrónica Industrial del Swiss Federal Institute of Technology, construyeron un prototipo de un vehículo de dos ruedas basado en un péndulo invertido llamado Joe, al cual le colocaron pesos en la varilla del péndulo para simular el peso de un ser humano en baja escala (Figura 2.2) [Grasser, 2002].



Figura 2.2 Joe

Sherman (2003), construyó el péndulo invertido sobre dos ruedas que tienen su centro de gravedad por encima de las ruedas llamado Balibot (Figura 2.3) [Sherman, 2003].

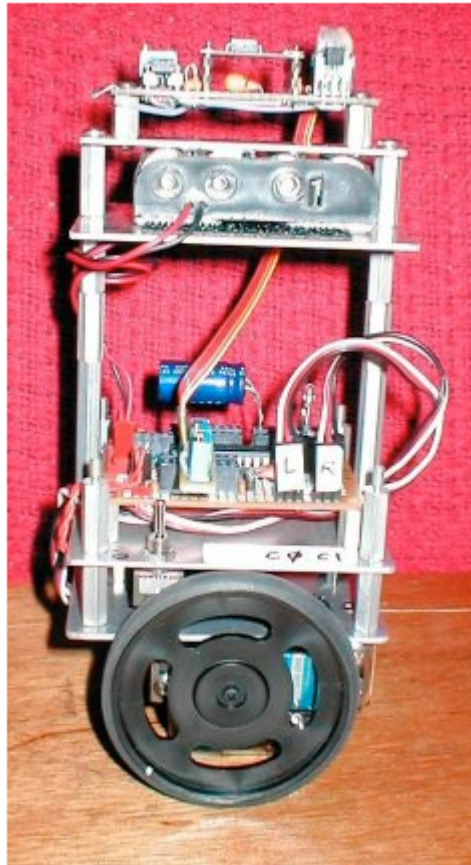


Figura 2.3 Balibot

Anderson (2003), construyó un robot de balanceo sobre dos ruedas denominado nBot. El robot utiliza el controlador del robot HC11 desarrollado para el MIT 6.270 Curso de Robótica, el controlador del robot mismo que se utiliza en la LegoBot y SR04. (Figura 2.4) [Anderson, 2003].



Figura 2.4 nBot

Ooi (2003), como proyecto de final de carrera en la escuela de Ingeniería Mecánica de la Universidad de Western Australia, realizó la construcción de un péndulo invertido sobre dos ruedas [Ooi, 2003].

También se han construido péndulos invertidos sobre dos ruedas utilizando la plataforma Lego Mindstorm, Hassenplug (2002), construyó un robot péndulo invertido sobre dos ruedas que constantemente intenta ajustar su punto de equilibrio, utilizando un acelerómetro para detectar la inclinación, llamado Legway [Hassenplug, 2002] (Figura 2.5).



Figura 2.5 Legway

Hurbain's (2007), construyó el NXTwat [Hurbains]. Watanabe (2007) (Figura 2.6), de la Universidad de Waseda en Japón, construyó el NXRway-G. Yamamoto (2008), construyó el NXTway-GS, el cual balancea y además desplaza el robot con la utilización de un control remoto [Watanabe, 2007].

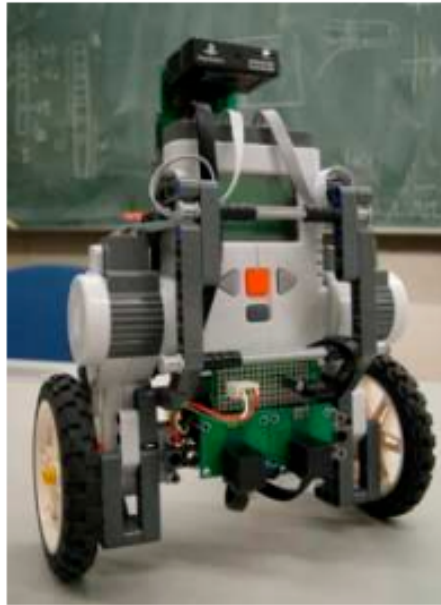


Figura 2.6 Watanabe

## **2.1 Control del péndulo invertido**

Los péndulos invertidos son una familia de dispositivos que constituyen un banco de pruebas muy completo e interesante para la ingeniería de control no lineal. El más estudiado de los miembros de esta familia es el denominado control invertido sobre un vehículo. Consiste en un péndulo o varilla que gira libremente por uno de sus extremos mediante una articulación situada sobre un carro que se mueve sobre una guía rectilínea horizontal bajo la acción de una fuerza  $F$ , que es la acción de control con la que se pretende actuar sobre la posición de la varilla.

## **2.2 Control PID**

Un PID, acrónimo de proporcional, iterativo y derivativo, es un mecanismo de control que se encarga de aplicar una acción correctora para disminuir la diferencia entre la salida de un sistema y la referencia que ésta

debería alcanzar, es decir, su error [Kuo, 1992]. Un PID, como indica su nombre, actúa en tres niveles diferentes:

- Proporcional.
  - Determina la reacción del error actual.
- Integral.
  - Corrige la integral del error para reducirlo a cero (error estacionario).
- Derivativo:
  - Determina la reacción del tiempo (derivada) en el que el error se produce.

La señal de control que se calcula mediante un PID viene dado por la siguiente expresión:

$$u(t) = K_p e(t) + K_i \int e(t) dt + K_d \frac{de(t)}{dt}$$

## 2.3 Control de ganancia programada

Los parámetros K se corresponden con las ganancias proporcional, integral y derivativa y son las que hay que calcular para obtener una señal de control adecuada. El uso de un control PID no garantiza un control óptimo ni la estabilidad del mismo, debido a que únicamente la respuesta del controlador se describe en términos del error medido. Debido a ello, en nuestro caso de estudio el control total estará compuesto de varios tipos de controladores, uno de ellos, un PID.

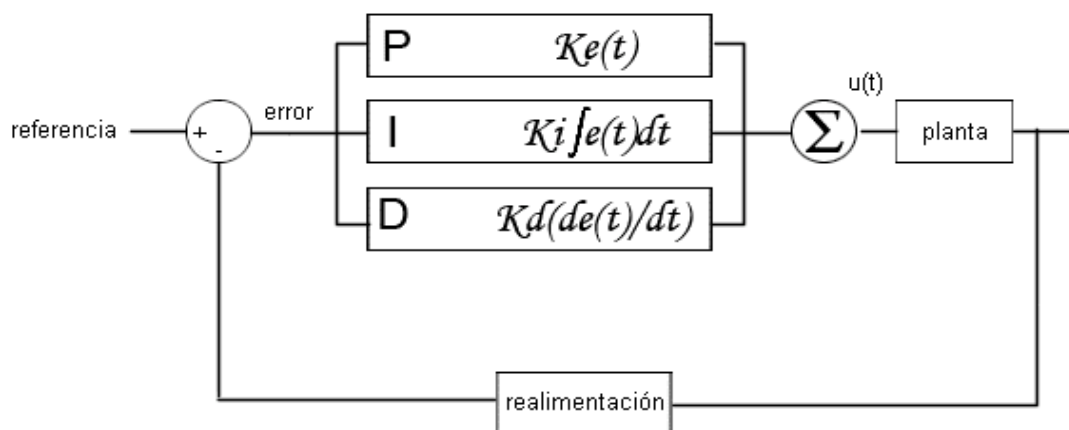


Figura 2.7 Esquema controlador PID

## 2.4 Regulador LQR

Uno de los principios a tener en cuenta a la hora de diseñar un mecanismo de control para un determinado problema es la realimentación [Ogata, 1997]. En la mayoría de los problemas de control el objetivo es lograr la estabilidad en torno a un punto de equilibrio, al que llamamos referencia, es decir, que la salida de nuestro sistema se mantenga dentro de unos valores que son impuestos.

Pero es imposible deducir que valores ha de tomar en cada instante la señal de control que va a hacer que nuestro sistema se estabilice sin tener en cuenta en qué estado se encuentra en ese momento. Por ello es necesario que los controles que se vayan a implementar puedan valorar que diferencia existe entre la referencia que queremos alcanzar y la situación actual en que nos encontramos. Esta técnica se conoce con el nombre de realimentación o retroalimentación, y en el problema que intentamos resolver se hace uso de la realimentación negativa o feedback negativo, pues lo que se pretende es conocer en cada instante cuán alejados estamos de nuestra señal de referencia (Figura 2.8).

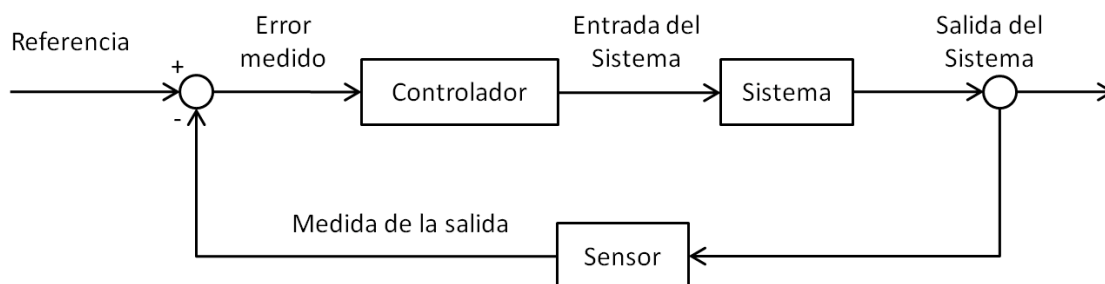


Figura 2.8 Esquema realimentación

Pero en ocasiones, dependiendo de la dificultad del problema, no solo basta con la propia retroalimentación, sino que es necesario algún mecanismo complementario que haga que la realimentación se realice de la forma más optimizada posible. Y es lo que vamos a conseguir haciendo uso de un regulador LQR (Figura 2.9).

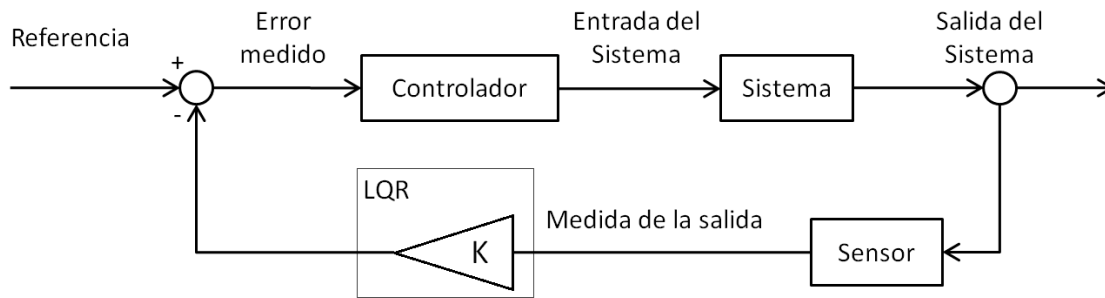


Figura 2.9 Esquema regulador LQR

## 2.5 Lógica Fuzzy

La lógica difusa se adapta mejor al mundo real en el que vivimos, e incluso puede comprender y funcionar con nuestras expresiones, del tipo "es muy grande", "es bastante alto", etc.

La clave de esta adaptación al lenguaje, se basa en comprender los cuantificadores de nuestro lenguaje (en los ejemplos de arriba "muy" y "bastante") [Zadeh, 1965].

Para cada conjunto difuso, existe asociada una función de pertenencia para sus elementos, que indican en qué medida el elemento forma parte de ese conjunto difuso. Las formas de las funciones de pertenencia más típicas son trapezoidales, lineales y curvas.

Se basa en reglas heurísticas de la forma SI (antecedente) ENTONCES (consecuente), donde el antecedente y el consecuente son también conjuntos difusos, ya sea puros o resultado de operar con ellos.

Los métodos de inferencia para esta base de reglas deben ser simples, veloces y eficaces. Los resultados de dichos métodos son un área final, fruto de un conjunto de áreas solapada entre sí (cada área es resultado de una regla de inferencia). Para escoger una salida concreta a partir de tanta premisa difusa, el método más usado es el del centroide, en el que la salida final será el centro de gravedad del área total resultante.

Los datos de entrada suelen ser recogidos por sensores, que miden las variables de entrada de un sistema. El motor de inferencias se basa en chips difusos, que están aumentando exponencialmente su capacidad de procesamiento de reglas año a año.

Un esquema de funcionamiento típico para un sistema difuso podría ser de la siguiente manera (Figura 2.9):





Figura 2.10 Funcionamiento de un sistema

En la figura, el sistema de control hace los cálculos con base en sus reglas heurísticas. La salida final actuaría sobre el entorno físico, y los valores sobre el entorno físico de las nuevas entradas serían medidas por sensores del sistema.

### 3 Construcción del Péndulo Invertido

Para la construcción de robot en configuración de péndulo invertido se ha usado la plataforma Lego Mindstorms NXT, ya que es un entorno de construcción y programación de robots de uso muy accesible y extendido, con una gran calidad y fiabilidad de motores, sensores y procesadores, que además cuenta con una gran comunidad de desarrolladores y multitud de entornos para trabajar.

El robot contará con el bloque programable NXT el cual contiene un microcontrolador ARM7 de 32 bits con memoria flash, es el elemento que contiene y ejecuta los programas realizados en un computador, y éstos permiten que un robot Mindstorms se mueva y pueda realizar diferentes operaciones.

Además se añadirá el sensor giroscópico NXT Gyro de Hitechnic (Figura 3.1), el cual detecta la velocidad angular y retorna un valor que representa el número de grados de rotación por segundo, de la misma forma que indica la dirección de rotación. El sensor Gyro puede medir rotación en una escala de  $\pm 360^\circ$ . Con él se pueden construir y controlar robots que se pueden balancear, donde la medición de la rotación es esencial. Además al robot se le añadirán sensores de color y ultrasonidos, que en este proyecto no serán utilizados, pero que se dejarán instalados en el robot a disposición de futuros estudios sobre el mismo.



Figura 3.1 Giróscopo NXT Gyro de Hitec

Para que el robot disponga de movilidad se instalarán en él dos servomotores de corriente DC contruidos en base a una gran cantidad de engranajes internos. Éstos pueden rotar hasta alcanzar 170 rpm, trabajan en un rango de -9V a +9V y tienen un peso de 80 gramos cada uno.

Con todos estos componentes del robot la configuración queda de la siguiente manera (Figura 3.2):



Figura 3.2. Robot LEGO NXT en configuración de péndulo invertido

Las características físicas, como la altura, la masa, el centro gravedad, etc... propias de esta configuración del péndulo y las propias de los otros elementos, como la potencia y el par de los motores, serán considerados en la construcción y simulación del modelo matemático incorporándolas a las ecuaciones para reflejar así un comportamiento lo mas ajustado posible al modelo real del robot.

### **3.1 Otras configuraciones del Lego NXT**

Podemos tener varias configuraciones del Lego NXT, con diferentes tipos de ruedas, centro de gravedad, colocación de los distintos sensores como el giroscopio, acelerómetro y otros muchos accesorios extras, por ejemplo ver Figura 3.3 y Figura 3.4



Figura 3.3 Ejemplo configuración I



Figura 3.4 Ejemplo configuración II

## 4 Modelado matemático del sistema

El primer paso necesario para el diseño de un controlador capaz de mantener estable el cuerpo del robot es encontrar un modelo del sistema, es decir, una representación abstracta de las características y propiedades del mismo. En nuestro caso, se pretende obtener un modelo matemático que refleje el comportamiento del sistema para su simulación, o lo que es lo mismo, un conjunto de ecuaciones matemáticas que relacionan una serie de variables con las características físicas del sistema. Con el modelo y un software matemático adecuado, en nuestro caso Matlab, podemos simular cuál es el comportamiento del sistema bajo cualquier condición. En definitiva, lo que se persigue es evaluar las salidas que proporciona el sistema cuando le son aplicadas unas determinadas entradas (figura 4.1), salidas que en nuestro caso serán de gran utilidad a la hora de diseñar el controlador.

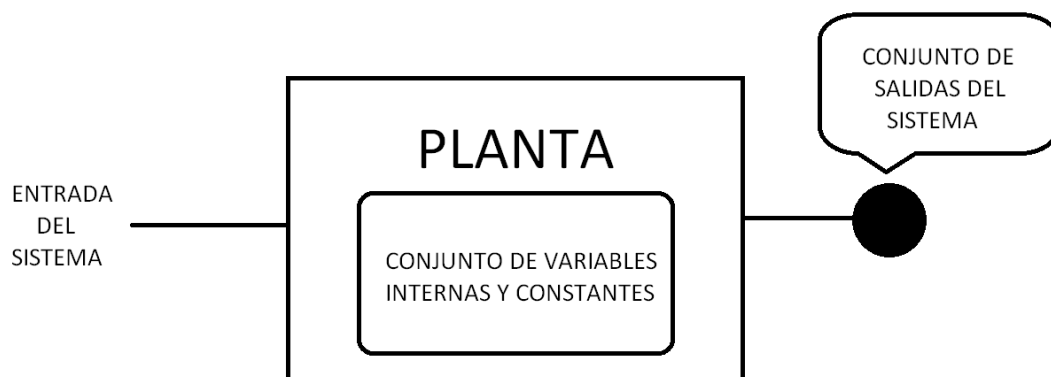


Figura 4.1 Esquema del modelo matemático

Una de las claves a la hora de implementar un control eficiente de cualquier sistema es partir de un modelo que refleje fielmente el comportamiento del sistema real. Sin embargo elaborar un modelo exacto es muy complicado y la mayoría de las veces imposible. Por tanto se busca establecer un equilibrio entre las complicaciones que conlleva elaborar el modelo con las ventajas que proporciona dicho modelo a la hora de su estudio. Por ejemplo, no interesa un modelo matemático demasiado elaborado o complicado si a la hora de su estudio y posterior simulación se obtienen controladores que apenas mejoran el comportamiento de la planta del que se hubiese obtenido a partir de modelos más simplificados.

## 4.1 Obtención del modelo matemático del péndulo invertido

El péndulo invertido es un problema muy habitual dentro del estudio de la ingeniería de control, con la diferencia de que el problema clásico consiste en un carro móvil sobre el que se encuentra un sólido que se pretende mantener en equilibrio. En nuestro caso partimos de un único cuerpo que incluye ruedas, motores, sensores y demás dispositivos, como se observa en la figura 4.2.

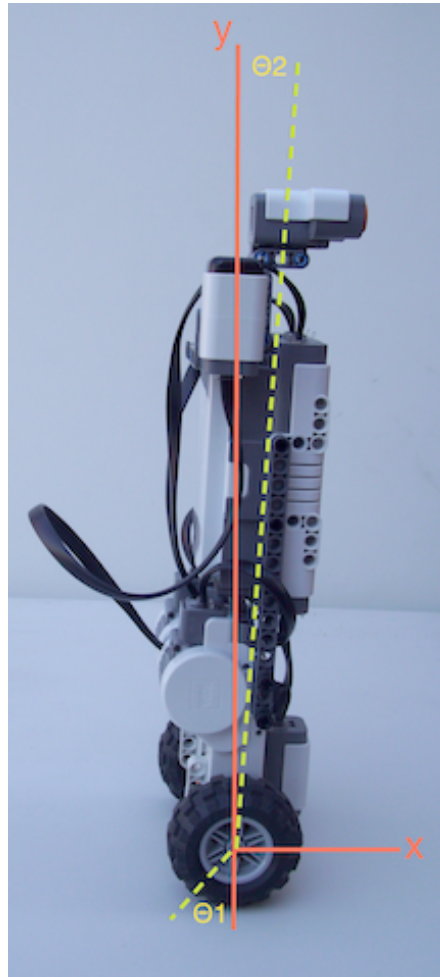


Figura 4.2 Fotografía del robot

Originalmente se puede considerar el problema del péndulo invertido como un sistema no lineal, pero la dificultad que implica el modelado de estos sistemas hace que sea necesaria una linealización del mismo. Todo sistema no lineal se puede linealizar dentro de un rango en el cuál se comporte se forma muy parecida a un sistema lineal. En el caso del péndulo invertido ese rango lo establecemos en torno al punto de equilibrio, es decir, el punto en que el robot se encuentra a cero grados con la vertical. Con ello, podemos simplificar





enormemente la obtención de un modelo matemático que nos es suficiente para la después implementación de los controles en la planta real.

Otro aspecto que nos va a ayudar a simplificar el esquema es que en el estudio matemático del mismo, se va obviar el hecho de la existencia de dos servomotores y únicamente se estudiará el sistema en base a una aplicación simultánea de los dos en paralelo.

#### 4.1.1 Parámetros del sistema

En primer lugar es necesario recopilar información sobre parámetros que influyen en el comportamiento del sistema, obtenidos de las especificaciones técnicas del fabricante, y que se muestran en la siguiente tabla:

Parámetros constantes del sistema		
Símbolo	Descripción	Valor
$J_0$	Momento de inercia del motor	0.000842 kg m <sup>2</sup>
$J_1$	Momento de inercia de las ruedas	0.00000625 kg m <sup>2</sup>
$J_2$	Momento de inercia del cuerpo	0.09 kg m <sup>2</sup>
R	Radio de las rueda	0.027 m
$M_1$	Masa de la rueda	0.029 kg
$M_2$	Masa del cuerpo	0.7 kg
L	Distancia del eje al centro de gravedad	0.15 m
b	Fricción del motor	0.00129 N m s
g	Aceleración de la gravedad	9.8 m/s <sup>2</sup>

Tabla 4.1 Parámetros constantes del sistema

Parámetros variables del sistema		
Símbolo	Descripción	Unidad
$\theta_1$	Ángulo de la rueda	rad
$\theta_2$	Ángulo del cuerpo	rad
$\dot{\theta}_1$	Velocidad angular de la rueda	rad/s
$\dot{\theta}_2$	Velocidad angular del cuerpo	rad/s
$\ddot{\theta}_1$	Aceleración de la rueda	rad/s <sup>2</sup>
$\ddot{\theta}_2$	Aceleración del cuerpo	rad/s <sup>2</sup>

Tabla 4.2 Parámetros variables del sistema

#### 4.1.2 Base matemática del sistema

Como vemos en la tabla 4.2, el modelo matemático sobre el que vamos a realizar las simulaciones y vamos a elaborar un controlador va a estar basado en seis variables: los ángulos correspondientes al cuerpo del robot y de las ruedas, así como de las velocidades y aceleraciones angulares asociados a ellos.  $\theta_2$  corresponde al ángulo del cuerpo que se puede obtener mediante la integración de la velocidad angular que devuelve el giróscopo,  $\dot{\theta}_2$ .  $\theta_1$  es el ángulo de avance de la rueda, y su valor se puede leer directamente de los encoders que los motores tienen instalados. Derivando este valor obtenemos la velocidad angular  $\dot{\theta}_1$ . Las segundas derivadas que aparecen en la tabla se corresponden con las aceleraciones que sufren las ruedas y el cuerpo del robot,  $\ddot{\theta}_1$  y  $\ddot{\theta}_2$  respectivamente.

Conocemos las siguientes ecuaciones [Stimac, 1999] [Novakowski, 2006]:

$$\dot{X}_1 = R\dot{\theta}_1 \quad (1)$$

$$M_1\ddot{X}_1 = F_R - F_H \quad (2)$$

$$J_1\ddot{\theta}_1 = RF_R \quad (3)$$

Donde la  $X_1$  hace referencia a la posición de las ruedas dentro del eje x propiamente dicho. En estas ecuaciones vemos la relación de la velocidad



(primera derivada) y la aceleración (segunda derivada) lineal de las ruedas con parámetros del robot. A saber:  $M_1$  es la masa de la rueda,  $F_R$  es la fricción entre las ruedas y el suelo,  $J_1$  se corresponde con los momentos de inercia de las ruedas,  $R$  es el radio de las ruedas y  $F_H$  representa la componente en el eje de abscisas de la fuerza que ejerce el cuerpo sobre las ruedas. Todos los valores se encuentran en la tabla 4.1.

Por otro lado conocemos [Stimac, 1999] [Novakowski, 2006]:

$$M_2 \ddot{X}_2 = F_H \quad (4)$$

$$M_2 \ddot{Z}_2 = F_V - M_2 g \quad (5)$$

$$J_2 \ddot{\theta}_2 = LF_V \sin \theta_2 - LF_H \cos \theta_2 + u \quad (6)$$

Al contrario que en el conjunto de ecuaciones anteriores, el subíndice "2" hace referencia al cuerpo del robot. Con estas ecuaciones quedan relacionadas la velocidad y aceleraciones dentro del eje x y z (caída del robot), con parámetros del mismo.  $F_V$  representa la componente vertical de la fuerza que ejerce el cuerpo sobre las ruedas,  $L$  es la distancia del eje de las ruedas al centro de gravedad,  $M_2$  es la masa del cuerpo del robot y  $J_2$  se corresponde con el momento de inercia del mismo. Los valores se encuentran reflejados en la tabla 4.1.

La relación entre la dinámica del cuerpo del robot y las ruedas es la siguiente:

$$X_2 = X_1 + L \sin \theta_2 \quad (7)$$

$$Z_2 = L \cos \theta_2 \quad (8)$$

Presuponiendo que lo que se busca es un valor mínimo del ángulo del cuerpo  $\theta_2$  (linealización del sistema), se puede recurrir a la siguiente aproximación:

$$\begin{aligned} \theta_2^2 &\approx 0 \\ \cos \theta_2 &\approx 1 \\ \sin \theta_2 &\approx \theta_2 \end{aligned}$$

Y mediante las ecuaciones anteriores podemos obtener las siguientes expresiones:

$$h_1\ddot{\theta}_1 + h_2\ddot{\theta}_2 + h_3\dot{\theta}_1 = h_4u \quad (9)$$

$$h_2\ddot{\theta}_1 + h_5\ddot{\theta}_2 + h_6\dot{\theta}_2 = u \quad (10)$$

Donde:

$$h_1 = (J_1 + J_0R^2 + (M_1 + M_2)R^2) \quad (11)$$

$$h_2 = M_2LR^2 \quad (12)$$

$$h_3 = bR^2 \quad (13)$$

$$h_4 = R \quad (14)$$

$$h_5 = J_2 + M_2L \quad (15)$$

$$h_6 = -M_2gL \quad (16)$$

La señal “u” representa el par motor aplicado a los servomotores eléctricos del robot, es decir, será la señal de control cuyo valor será calculado por los controladores que se van a diseñar a continuación y que servirán para mantener al robot en estado de equilibrio.

#### 4.1.2 Modelo matemático en el espacio de estados

Una vez tenemos las ecuaciones matemáticas que nos modelan el comportamiento físico de nuestro sistema, el siguiente paso a realizar es transformarlas al espacio de estados<sup>3</sup>.

El problema del péndulo invertido sigue un comportamiento dinámico<sup>4</sup> y continuo<sup>5</sup>, por ello el esquema matemático que lo modela sí debe ser dinámico. Esto quiere decir que es necesario algún mecanismo que nos facilite la

---

<sup>3</sup> **Espacio de estados:** una representación de espacio de estados es un modelo matemático de un sistema físico descrito mediante un conjunto de entradas, salidas y variables de estado relacionadas por ecuaciones diferenciales de primer orden que se combinan en una ecuación diferencial matricial de primer orden.

<sup>4</sup> **Sistema dinámico** es un sistema cuyo estado evoluciona con el tiempo. El comportamiento en dicho estado se puede caracterizar determinando los límites del sistema, los elementos y sus relaciones; de esta forma se puede elaborar modelos que buscan representar la estructura del mismo sistema.

<sup>5</sup> **Sistema continuo:** son sistemas cuyas variables evolucionan de forma continua en el tiempo. Estos sistemas pueden en general modelarse mediante Ecuaciones Diferenciales.

recopilación de información sobre la evolución que ha seguido el robot anteriormente. Por ejemplo, en un determinado instante, si nos encontramos el robot inclinado un cierto número de grados respecto a la vertical es muy difícil, prácticamente imposible, deducir que valor debemos dar a nuestra señal de control “u”, que como hemos dicho es el par motor a aplicar en los servomotores. La razón reside en que hay que tener en cuenta no sólo el ángulo de inclinación que tiene el robot en ese instante, sino otras variables como la velocidad angular con la que está cayendo o qué velocidad teníamos aplicada a las ruedas anteriormente, entre otros aspectos. No es la misma situación la que representa al robot inclinado un determinado número de grados intentando mantenerse erguido, que una situación similar pero producida por una perturbación ajena al mismo (como puede ser un pequeño empujón), o por un cambio en el desnivel del terreno. Por ello, en la toma de decisiones a la hora de elaborar los controladores hemos de tener en cuenta no sólo un número concreto de variables, sino qué valor tenían anteriormente, y es aquí donde reside el concepto de “estado del sistema”.

El estado que representa en qué condiciones se encuentra el robot en un determinado instante viene descrito por cuatro variables, que son: el ángulo del cuerpo del robot con respecto a la vertical ( $\theta_2$ ), la velocidad angular del cuerpo ( $\dot{\theta}_2$ ), el ángulo de las ruedas ( $\theta_1$ ) y su velocidad angular ( $\dot{\theta}_1$ ).

Es necesario pues, expresar las salidas del sistema en función del estado y del valor de las entradas:

$$\dot{\theta}(t) = f(x(t), u(t)) \quad (17)$$

$$y(t) = h(x(t), u(t)) \quad (18)$$

En modelado la representación del espacio de estados, las variables vienen dadas como vectores y las ecuaciones algebraicas en forma matricial, de ahí la modificación de las ecuaciones (17) y (18) en:

$$\dot{x} = Ax + Bu \quad (19)$$

$$y = Cx + Du \quad (20)$$

La ecuación (19) nos calcula el estado siguiente (derivado) a partir de la señal “u” y el estado actual, mientras que la ecuación (20) nos sirve para obtener las salidas pudiendo seleccionar las variables de estado que consideremos oportunas.



Teniendo en cuenta nuestras variables de estado  $x = (x_1, x_2, x_3, x_4) = (\theta_2, \dot{\theta}_2, \theta_1, \dot{\theta}_1)$ , podemos derivar las ecuaciones de estado de la siguiente forma:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} \dot{\theta}_2 \\ \ddot{\theta}_2 \\ \dot{\theta}_1 \\ \ddot{\theta}_1 \end{bmatrix} \begin{bmatrix} x_2 \\ a_1 x_1 + a_2 x_4 + b_1 u \\ x_4 \\ a_3 x_1 + a_4 x_4 + b_2 u \end{bmatrix}$$

Es por tanto un sistema de dimensión 4, ya que  $\dim(x) = 4$ , donde:

$$a_1 = \frac{-h_1 h_6}{h_5 h_1 - h_2^2}$$

$$a_2 = \frac{h_2 h_3}{h_5 h_1 - h_2^2}$$

$$a_3 = \frac{h_2 h_6}{h_5 h_1 - h_2^2}$$

$$a_4 = \frac{-h_3 h_5}{h_5 h_1 - h_2^2}$$

$$b_1 = \frac{h_1 - h_2 h_4}{h_5 h_1 - h_2^2}$$

$$b_2 = \frac{-h_2 - h_5 h_4}{h_5 h_1 - h_2^2}$$

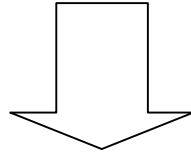
Donde  $h_1, h_2, h_3, h_4, h_5, h_6$  se obtienen de las ecuaciones (11) – (16).



Con ello, ya podemos representar el modelo en el espacio de estados de la siguiente forma:

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$



$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ a_1 & 0 & 0 & a_2 \\ 0 & 0 & 0 & 1 \\ a_3 & 0 & 0 & a_4 \end{bmatrix} x + \begin{bmatrix} 0 \\ b_1 \\ 0 \\ b_2 \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} u$$

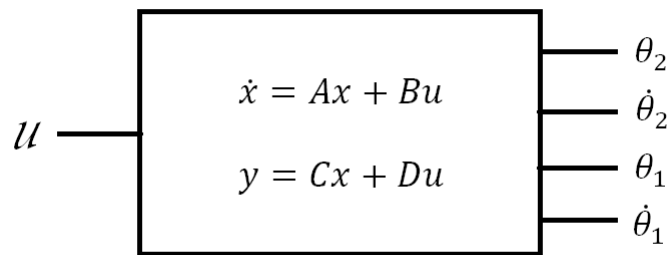


Figura 4.3 Esquema del modelo matemático en espacio de estados

Como representa la figura 4.3, las salidas del sistema modelado son calculadas mediante una única entrada correspondiente a la señal de control y al estado en que se encuentre.

## 4.2 Simulación del modelo

Con el objetivo de comprobar que la planta modelada a partir del sistema real representa fielmente el comportamiento del mismo, se procede a simular en Matlab el sistema en lazo abierto. Teóricamente, se busca comprobar que la excitación del sistema provoca la inestabilidad esperada causada por la ausencia de ningún mecanismo de control que lo estabilice.

El primer paso es trasladar a un lenguaje de programación adecuado las matrices  $A$ ,  $B$ ,  $C$  y  $D$  que representan el espacio de estados, junto con los parámetros del mismo, es decir las constantes y ecuaciones de la sección 4.1.1 y 4.1.3.

Se crea el diagrama de bloques donde se representa el sistema en lazo abierto y que se observa en la figura 4.4. Este diagrama se compone del módulo que simboliza la planta, mas una entrada escalón que nos servirá para simular una perturbación a la que se someta al robot de la magnitud que se considere conveniente.

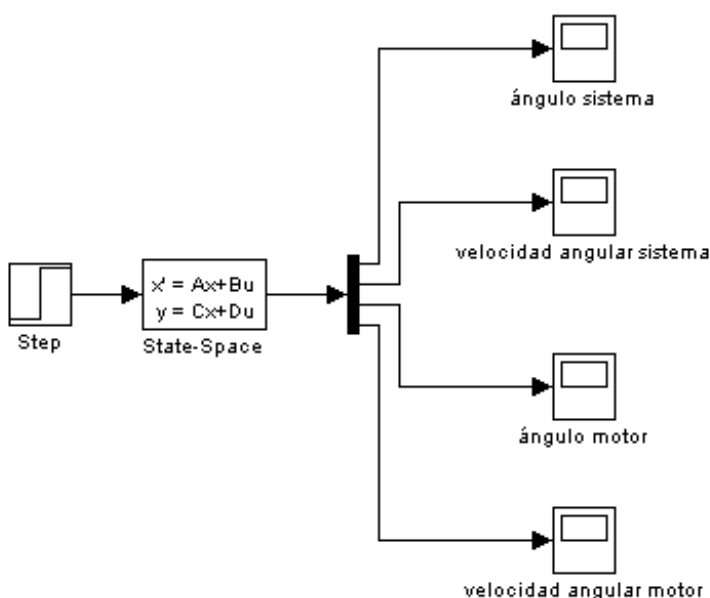


Figura 4.4 Diagrama de bloques del sistema en lazo abierto

Como se ha descrito, cualquier pequeña perturbación por mínima que sea hace inestable el sistema. Obviamente si éste no es alterado, obtenemos un comportamiento ideal del mismo. En nuestro caso, al perseguir la estabilidad vertical del robot, el comportamiento ideal es un ángulo de inclinación nulo del cuerpo del robot. Pero es conocido que las condiciones que rodean al robot alteraran este comportamiento ideal.

Por ello, si el sistema recibe una alteración observamos el siguiente resultado:

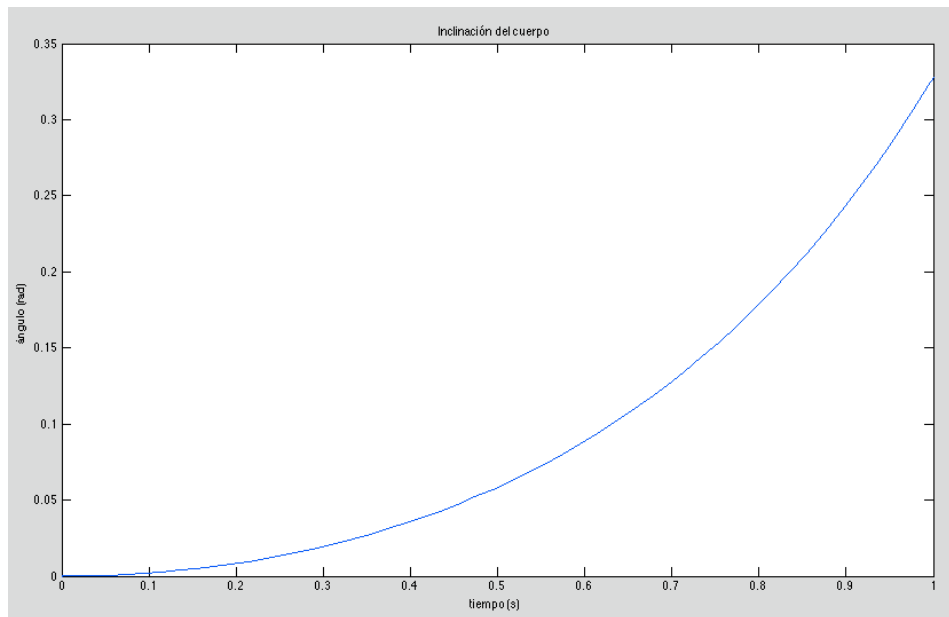


Figura 4.5 Gráfica de inclinación de cuerpo del robot. Simulación en lazo abierto

La gráfica correspondiente a la figura 4.5 nos muestra la evolución del ángulo de inclinación del cuerpo del robot después de aplicar una perturbación (step) de 0.08 a partir del instante 0, lo que se definió en las ecuaciones matemáticas como  $\theta_2$ . Como se observa, el ángulo del cuerpo del robot comienza a aumentar con forma exponencial, es decir, cae hasta colisionar con el suelo y cada vez a más velocidad, ya que como se ha dicho, no hay controlador que se encargue de estabilizarlo.

De esta forma damos por válido el modelo que se ha obtenido para el estudio del comportamiento del robot, y será a partir de éste sobre el que se diseñarán los diversos controladores que se van a implementar.

## 5 Diseño del control del péndulo

### 5.1 Regulador LQR

En nuestro caso, como vimos en la sección 2.4, la referencia que se persigue es clara, mantener el robot en un punto de equilibrio; y los valores con los que podemos realimentar el sistema también, son las mediciones que podemos hacer mediante los sensores que alberga el robot, es decir, el giroscopio<sup>6</sup> y los encoders<sup>7</sup>. Con ellos podemos obtener, bien directamente o mediante sencillas operaciones matemáticas, los ángulos de ruedas y del cuerpo del robot y sus respectivas velocidades angulares, lo que constituyen a su vez las variables de estado del modelo matemático que representa el problema.

A grandes rasgos un regulador LQR, del inglés “*linear quadratic regulator*”, es un dispositivo que proporciona una minimización de la suma de los esfuerzos de control, optimizándolos, y por tanto disminuyendo las desviaciones de la señal de un valor deseado. Además su uso es propicio en el trabajo con modelos basados en ecuaciones de estados como es nuestro caso. En la práctica, el regulador consiste en la asignación de unos determinados pesos a las variables con las que vamos a realimentar nuestra planta. Estos valores por los que multiplicamos las señales, llamados pesos, se obtienen de una serie de operaciones algebraicas. En primer lugar, se busca una ley de control que minimice:

$$J(x,u) = \int_0^{\infty} (x^T(t)Qx(t) + Ru^2(t))dt$$

Lo que pretendemos encontrar es una señal óptima de la siguiente forma:

$$u^{opt}(t) = -k_{opt}x(t) = -R^{-1}B^T Px(t)$$

Donde las matrices Q y R son parámetros especificados por el diseñador y la matriz P se obtiene de la siguiente ecuación algebraica de Riccati:

$$A^T P + PA - PBR^{-1}B^T P + Q = 0$$

---

<sup>6</sup> **Giroscopio:** es un dispositivo mecánico formado esencialmente por un cuerpo con simetría de rotación que gira alrededor de su eje de simetría. Cuando se somete el giroscopio a un momento de fuerza que tiende a cambiar la orientación del eje de rotación cambia de orientación en una dirección perpendicular a la dirección “intuitiva”.

<sup>7</sup> **Encoder:** codificador rotatorio, también llamado codificador del eje o generador de pulsos, usado para convertir la posición angular de un eje a un código digital, lo que lo convierte en una clase de transductor.





Por tanto, según los valores que se dé a las matrices  $Q$  y  $R$  hará variar los resultados. Dichas matrices reciben el nombre de matriz de coste ( $Q$ ), y matriz de rendimiento ( $R$ ). En general,  $Q$  tiene la forma de la matriz identidad sustituyendo los unos por diferentes pesos y el objetivo es realizar pruebas con diferentes valores para  $Q$  hasta encontrar la  $K$  que proporciona el control más eficiente. Y por otro lado,  $R$  adoptará la matriz unitaria  $1$ .

Lo que vamos a conseguir en nuestro caso es calcular las ganancias óptimas para cada variable de estado que son usadas en la realimentación. Para obtener dichos valores se realizarán varias pruebas para varios valores de  $Q$ . La implementación de todo ello está descrita en el punto X de la memoria correspondiente a la simulación con Matlab.

## **5.2 Controlador PID de ganancia programada**

Un PID, como indica su nombre, actúa en tres niveles diferentes Proporcional, Integral y Derivativo (sección 2.3).

En nuestro caso, aplicamos un controlador PID al problema del péndulo invertido, el cual es un problema matemático no lineal. En un principio sobre la estructura básica de un control PID se implementa el código para construir una rutina que garantice la estabilidad del robot en estático sin contemplar un desplazamiento del robot. Esta rutina se parametriza con los datos obtenidos en la sintonización del PID en la simulación del problema (apartado 6.1.5).

Partiendo de esta base, para garantizar la estabilidad del robot en condiciones que comprometan más dicha estabilidad, se ha pasado a implementar un controlador PID de ganancia programada, en el que los valores de las ganancias de los componentes proporcional, integral y derivativa evolucionan en función de la situación del robot. Se podría ver como la presencia de diferentes controladores, cada uno con sus correspondientes ganancias donde cada uno de ellos actuaría en función de las circunstancias que envuelvan al robot en un determinado instante.

De esta forma, se han tomado en cuenta tres de las cuatro variables del sistema para programar dichas ganancias, tomaremos en cuenta el ángulo de inclinación del cuerpo del robot, la velocidad angular del balanceo, y la velocidad angular de la rueda, para determinar las ganancias, y no consideraremos la posición de la rueda (su ángulo).

Se pretende tener tres zonas de actuación del PID (Figura 5.2), una zona "A" de trabajo en las que las condiciones de estabilidad no se vean prácticamente amenazadas, la cual tendrá los valores de las ganancias calculados en la sintonía del PID básico. Otra zona "B" en la que la situación del robot es más extrema, y que requerirá de una acción de control más drástica y rápida (con una ganancia derivativa mayor, por lo tanto). Y por último, una zona "C" en la que las acciones de control deberán ser extremas para compensar la situación del robot, cuya estabilidad se verá muy comprometida por su inclinación, balanceo y velocidad de las ruedas.

Ejemplo zona de trabajo "A":

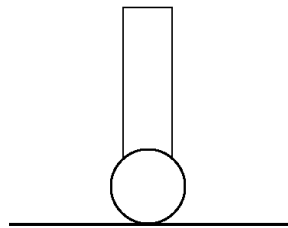


Figura 5.2.1 Zona A

Ejemplo zona de trabajo "B":

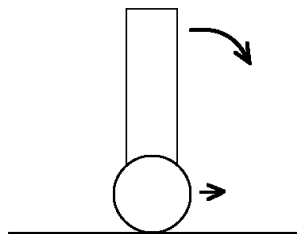


Figura 5.2.2 Zona B

Ejemplo zona de trabajo "C":

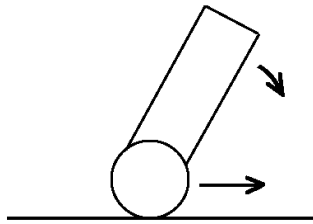


Figura 5.2.3 Zona C

Para poder definir tales zonas es necesario establecer unos límites para los valores de las señales que se van a tener en cuenta, hay que fijar unos rangos para saber qué valores de las ganancias del PID son aplicables cada momento. Por ejemplo, hemos de clasificar el ángulo que presenta el cuerpo del robot, para poder decidir que acción hay que tomar de cara a lograr la estabilidad del robot; un ángulo de inclinación muy elevado, que puede venir dado por una perturbación ajena al mismo, necesita de una respuesta de los motores más rápida y de mayor potencia pues la respuesta debe ser inmediata. De modo contrario, con el robot estabilizado, las pequeñas variaciones que pueden producirse en las mediciones de la inclinación no pueden ser solventadas de la misma forma a la anterior descrita pues con acciones drásticas sobre los motores podemos conseguir desestabilizar el robot o llegar incluso a situaciones de periodicidad donde el robot pretende corregir una inestabilidad que él mismo ha producido al intentar compensar la deriva.

Pero no sólo hemos de tener en cuenta la inclinación del robot para inducir las zonas de trabajo, también han de conocerse la velocidad con la que se produce esa inclinación y que velocidad tienen las ruedas.

La forma en la que interaccionan entre sí las tres variables que vamos a tener en cuenta a la hora de situar el PID en una zona de trabajo de las tres que hemos definido se van a implementar mediante lógica borrosa o lógica fuzzy. Dependiendo de los valores que adopten cada una de ellas y con unas reglas que será necesario especificar, se podrá inferir por tanto el conjunto de valores para las ganancias proporcional, integral y derivativa que hay que aplicar al controlador PID en cada momento.



### 5.3 Control basado en Lógica borrosa

Al robot se le suministra una única señal de control encargada de controlar la potencia suministrada a los motores para compensar el desequilibrio del cuerpo. Esta señal la calcula el controlador PID y depende de los pesos que calcula el LQR para las cuatro variables del sistema y de las ganancias de las partes proporcional, integral y derivativa del controlador [Kosko, 1992].

Vamos dividir la zona de trabajo del controlador PID en tres regiones A, B y C con valores específicos para las ganancias del PID en cada una de ellas. Esas zonas se seleccionarán mediante lógica borrosa en función de tres de las cuatro variables del sistema: la inclinación del robot, la velocidad de balanceo y la velocidad de las ruedas. En cada una de ellas la señal de control debe ser lo suficientemente rápida y enérgica para mantener el equilibrio del robot en ese momento.

Definimos por tanto tres conjuntos de ganancias ( $K_p$ ,  $K_i$ ,  $K_d$ ) para tres regiones de actuación del PID. Una primera región de trabajo A (con sus correspondientes  $K_{pA}$ ,  $K_{iA}$ ,  $K_{dA}$ ) en la que la estabilidad del robot no está comprometida, no existe apenas inclinación del cuerpo, y las velocidades de balanceo y de las ruedas son pequeñas. Definimos otra región de trabajo B del PID (también con sus correspondientes  $K_{pB}$ ,  $K_{iB}$ ,  $K_{dB}$ ) en la que el equilibrio necesita de una acción más específica del controlador ya que hay más inclinación del cuerpo del robot, mayor error, y las velocidades de balanceo y de las ruedas son más elevadas. Por último se define otra zona de trabajo C del controlador (con  $K_{pC}$ ,  $K_{iC}$ ,  $K_{dC}$  específicas) en la que, por lo acusado de la inclinación y las altas velocidades, el robot necesita de acciones de control más drásticas para mantener su equilibrio. [Zadeh, 1965] [Kosko, 1992]

En la implementación del controlador del robot se usará la lógica borrosa para seleccionar el grado de participación de cada una de las regiones en que se dividen las ganancias del PID.

La lógica borrosa se va a encargar de seleccionar las ganancias para estas partes proporcional, integral y derivativa del controlador, y las va a calcular en función de tres de las variables del sistema, el ángulo del robot, la velocidad angular del cuerpo, y la velocidad angular de la rueda.

Por tanto el sistema de lógica borrosa dependerá de tres entradas, el ángulo de inclinación de cuerpo de robot " $\theta$ ", la velocidad angular de balanceo del cuerpo del robot " $\dot{\theta}$ " y la velocidad angular de las ruedas del robot " $\dot{y}$ ", y devolverá tres salidas A, B y C que codifican el valor de los parámetros de

sintonía ese momento, y que serán usadas en el cálculo de la ley de control del PID.

Para la implementación del sistema borroso se utilizará el entorno Matlab, que genera un fichero .fis, que después exportamos a código C y que será integrado en el archivo de control, compilado y cargado en el brick NXT.

El entorno Matlab permite configurar el sistema de entradas, salidas y reglas de forma muy gráfica e intuitiva. Las entradas y salidas del sistema borroso quedan dispuestas como aparece en la siguiente figura 5.3.1.

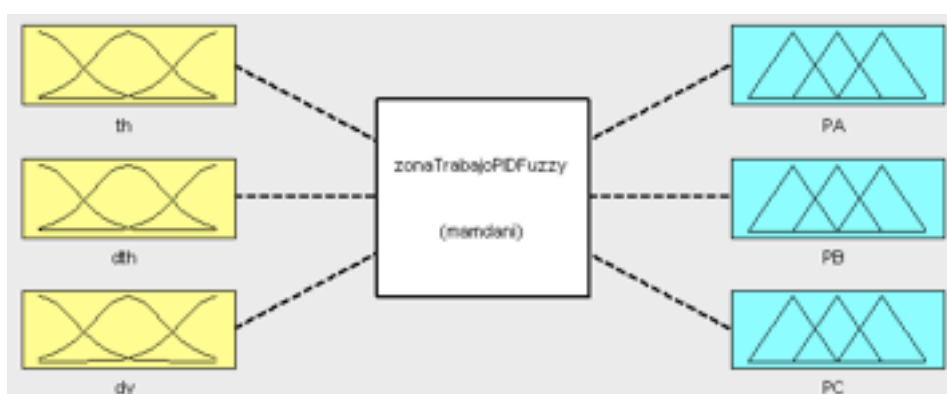


Figura 5.3.1 Entradas y salidas fuzzy

Definimos los rangos de cada una de estas entradas, y para cada una de ellas los etiquetamos valores "bajo", "medio" y "alto" y fijamos la forma de su función de pertenencia.

Para la entrada del ángulo  $\theta$  del cuerpo del robot acotaremos su valor entre 0 y 30 grados y la definimos como se observa en la figura 5.3.2.

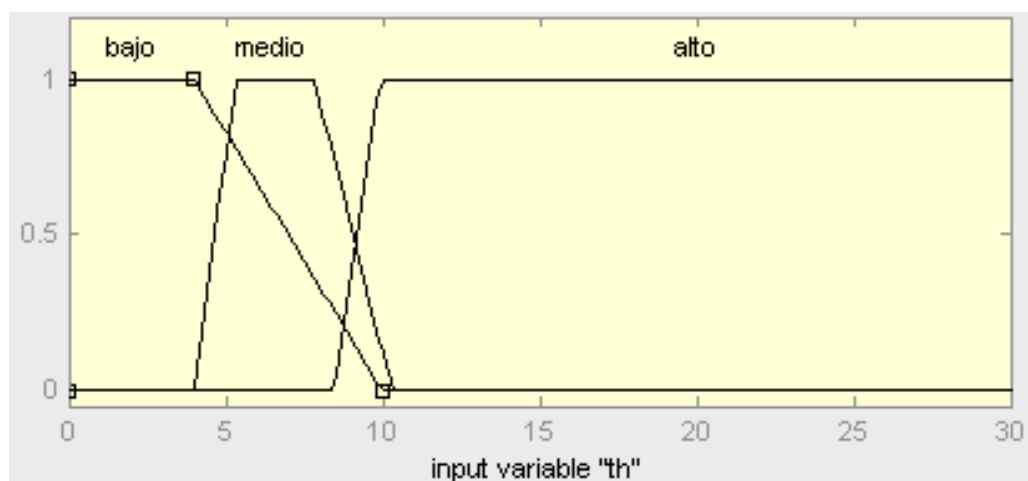


Figura 5.3.2 Entrada del ángulo del robot

Para la entrada de velocidad del balanceo del robot,  $d\theta$ , en grados/segundo (unidad con la que trabaja el giroscopio incorporado al robot) definimos un rango de discurso entre  $[0, 100]$  para sus valores y fijamos la forma de su función de pertenencia como sigue (Figura 5.3.3):

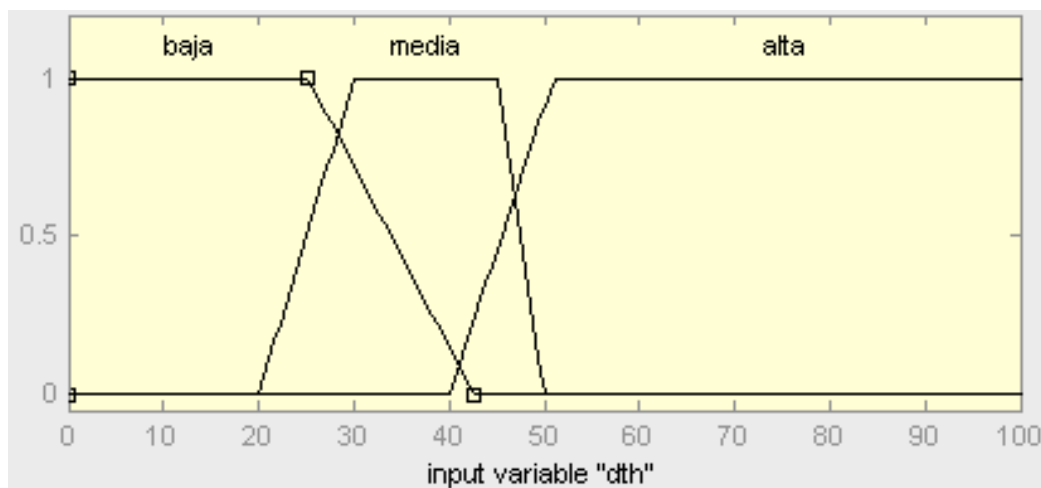


Figura 5.3.3 Entrada velocidad angular

Para la entrada de la velocidad de las ruedas del robot,  $dy$ , trabajaremos en unidades de radianes/segundo en un rango  $[0, 7]$ , y fijamos la forma de su función de pertenencia como aparece en la siguiente figura 5.3.4.

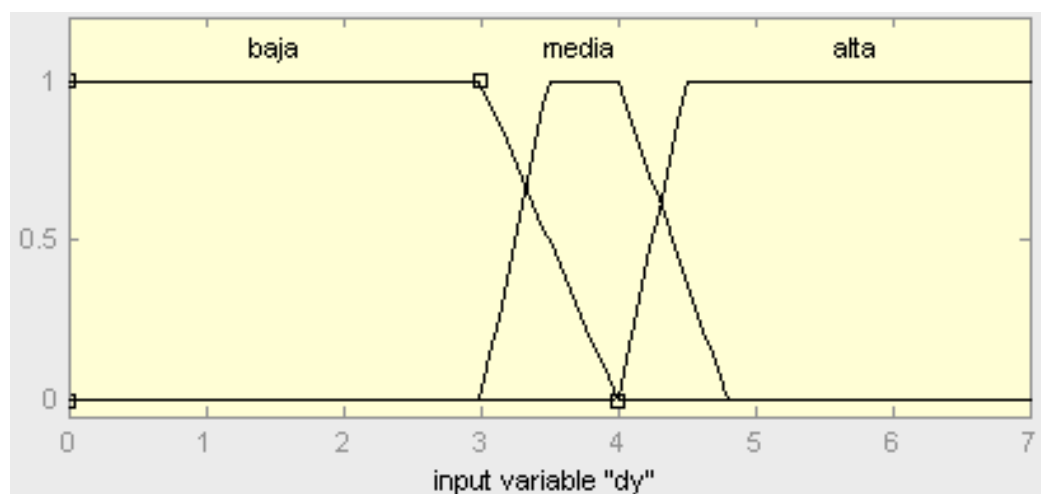


Figura 5.3.4 Velocidad de la rueda

Por último, cada una de las salidas están en el rango  $[0, 1]$ , con tres conjuntos fuzzy "bajo", "medio" y "alto" (Figura 5.3.5) de forma que así podamos calcular las ganancias:

$$k_p = Ak_{pA} + Bk_{pB} + Ck_{pC}$$

$$k_i = Ak_{iA} + Bk_{iB} + Ck_{iC}$$

$$k_d = Ak_{dA} + Bk_{dB} + Ck_{dC}$$

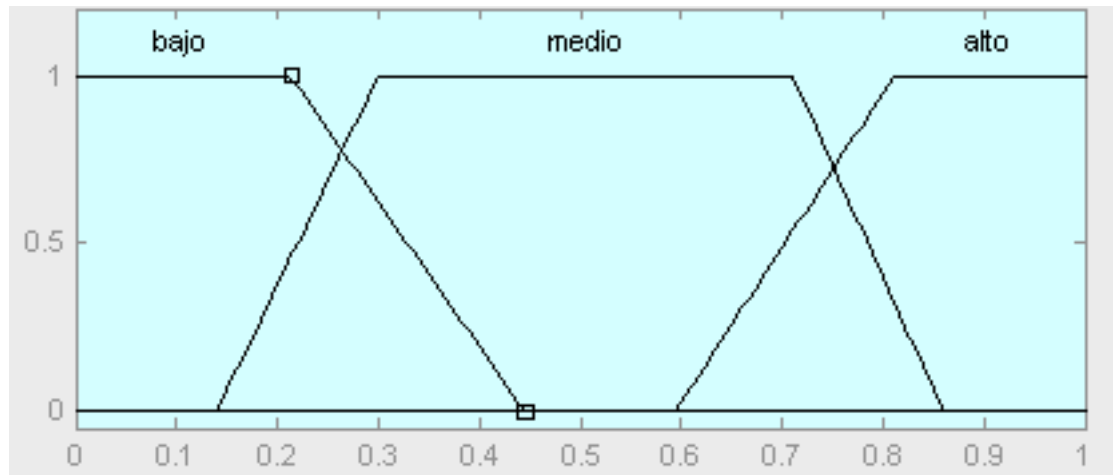


Figura 5.3.5 Formas sistemas borroso

Las reglas que se generan para el control de la zona de trabajo deben tener en cuenta las diferentes situaciones físicas en las que puede encontrarse el robot, y tener como antecedente las tres variables. Por ejemplo, para una situación ideal en la que el robot esté en equilibrio con un ángulo bajo respecto a la vertical y unas velocidades lineal y angular bajas requerirá una acción de control moderada en la zona A, pero si el robot tiene un cierto ángulo de inclinación, que se está compensando con un desplazamiento lineal, sin velocidad de balanceo del cuerpo, la acción de control corresponderá también a esta zona A moderada del control.

En base a todo esto, se definen las reglas de control fuzzy como sigue en la figura 5.3.6:

1. If (th is bajo) and (dth is baja) and (dy is not alta) then (PA is alto)(PB is bajo)(PC is bajo) (1)
2. If (th is bajo) and (dth is media) then (PA is medio)(PB is medio)(PC is bajo) (1)
3. If (th is bajo) and (dth is alta) then (PA is medio)(PB is medio)(PC is medio) (1)
4. If (th is medio) and (dth is baja) and (dy is media) then (PA is alto)(PB is bajo)(PC is bajo) (1)
5. If (th is medio) and (dth is media) then (PA is bajo)(PB is alto)(PC is bajo) (1)
6. If (th is medio) and (dth is alta) then (PA is bajo)(PB is medio)(PC is medio) (1)
7. If (th is alto) and (dth is baja) and (dy is alta) then (PA is alto)(PB is bajo)(PC is bajo) (1)
8. If (th is alto) and (dth is media) then (PA is medio)(PB is medio)(PC is medio) (1)
9. If (th is alto) and (dth is alta) then (PA is bajo)(PB is bajo)(PC is alto) (1)

Figura 5.3.6 Sistema de reglas

En cuanto a la función de conversión de un valor fuzzy a uno nítido, utilizada para obtener las salidas, se ha escogido la función "Bisector"<sup>8</sup>, la cual calcula la línea vertical que divide la región en dos sub-regiones de igual área.

Una vez definidas las entradas y salidas del sistema fuzzy, con sus rangos y funciones de pertenencia, las reglas del sistema, y la función de defuzzyficación podemos observar las superficies de control para la salida PA en la siguiente figura 5.3.7:

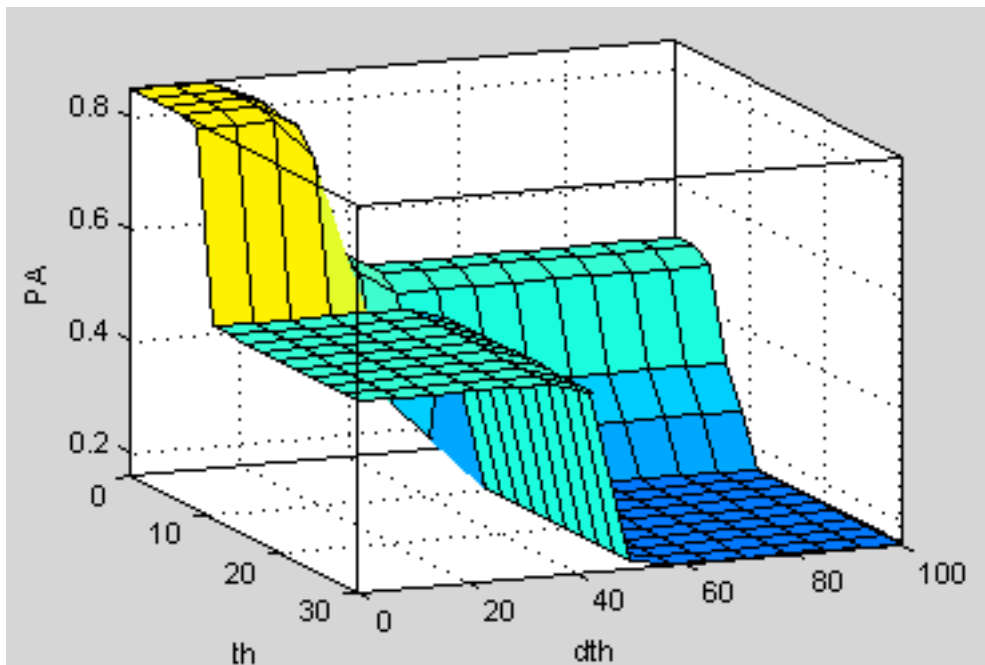


Figura 5.3.7 Superficie de Control zona A

En la siguiente imagen (Figura 5.3.8) podemos observar la superficie de control generada para la salida PB:

<sup>8</sup> <http://www.mathworks.es/products/fuzzy-logic/demos.html?file=/products/demos/shipping/fuzzy/defuzzdm.html>



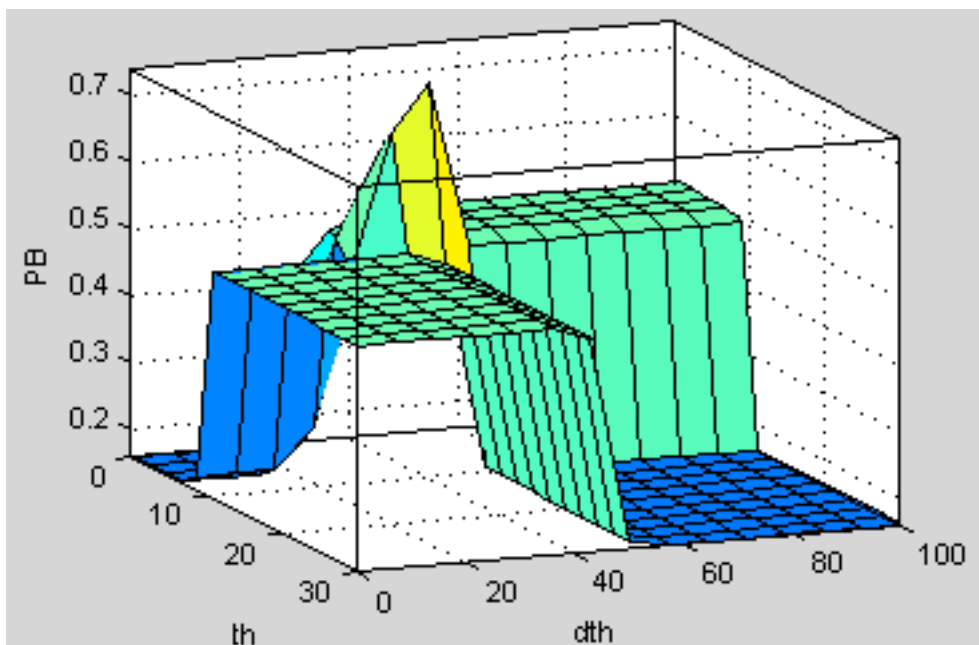


Figura 5.3.8 Superficie de Control zona B

Por último la superficie de control generada para la salida PC del sistema borroso puede observarse en la siguiente figura 5.3.9:

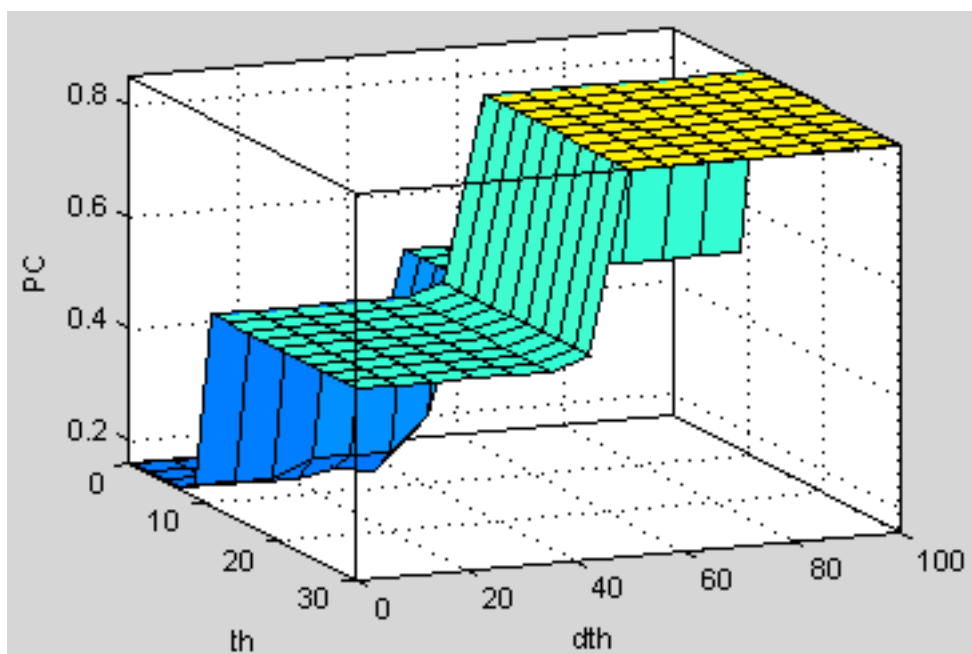


Figura 5.3.9 Superficie de Control zona C

Este es el diseño de controlador borroso para las zonas de trabajo del PID programado que se aplicará en el modelo del péndulo invertido, la implementación de dicho controlador se muestra en el capítulo 6 de este documento.

## 6 Simulación e Implementación

### 6.1 Simulación de los controladores

La simulación de los controladores, del mismo modo que se utilizó en la comprobación del modelo en la sección 4.1.2 de la memoria, se va a realizar con Matlab y su complemento Simulink<sup>9</sup>. El desarrollo de este apartado consistirá en analizar los resultados de simular el sistema aplicando los controladores estudiados en la sección anterior, y se llevará a cabo de forma incremental, desde el sistema en lazo cerrado hasta la inclusión del controlador Fuzzy para obtener las ganancias del PID de ganancia programada.

#### 6.1.1 Simulación del sistema en lazo cerrado

Se sabe que la realimentación de por sí es un mecanismo de control, pues la entrada de control que recibe la planta pasa a ser error, la diferencia entre la referencia de equilibrio y la medición del ángulo de inclinación. Se trata de una retroalimentación negativa o feedback negativo. En nuestro caso, la planta tiene cuatro variables de estado que son: ángulo y velocidad angular del sistema, ángulo y velocidad angular de las ruedas. Se procede a la realimentación de la planta con la suma de todas ellas, figura 6.1.

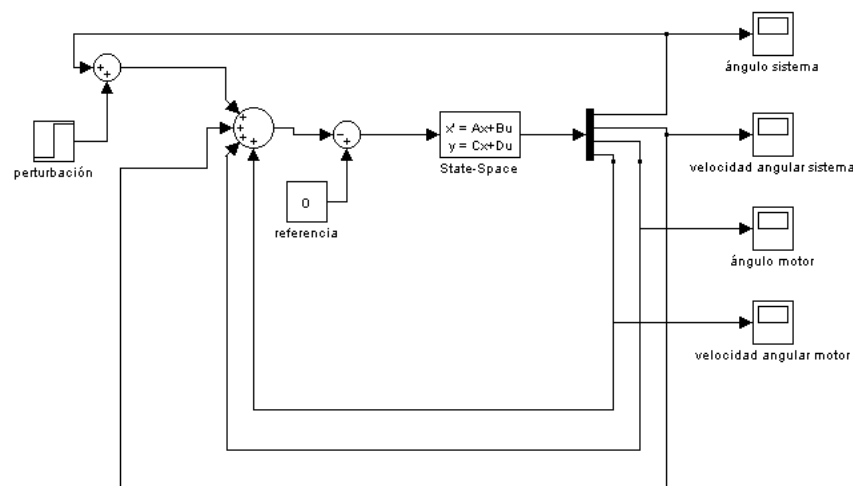


Figura 6.1 Diagrama de bloques del sistema en lazo cerrado

<sup>9</sup> **Simulink:** es un entorno de programación visual, que funciona sobre el entorno de programación Matlab. Entorno de programación de más alto nivel de abstracción que el lenguaje interpretado Matlab, simulink genera archivos con extensión .mdl

La perturbación va a venir dada por una alteración en el ángulo de inclinación del robot, que pretende emular un pequeño golpe o empujón, y que se va a sumar al ángulo que tiene el cuerpo en ese instante. En este caso vamos a incluir una perturbación en de 10 grados para observar cómo se comporta. Como referencia fijamos cero grados, la situación de equilibrio del sistema.

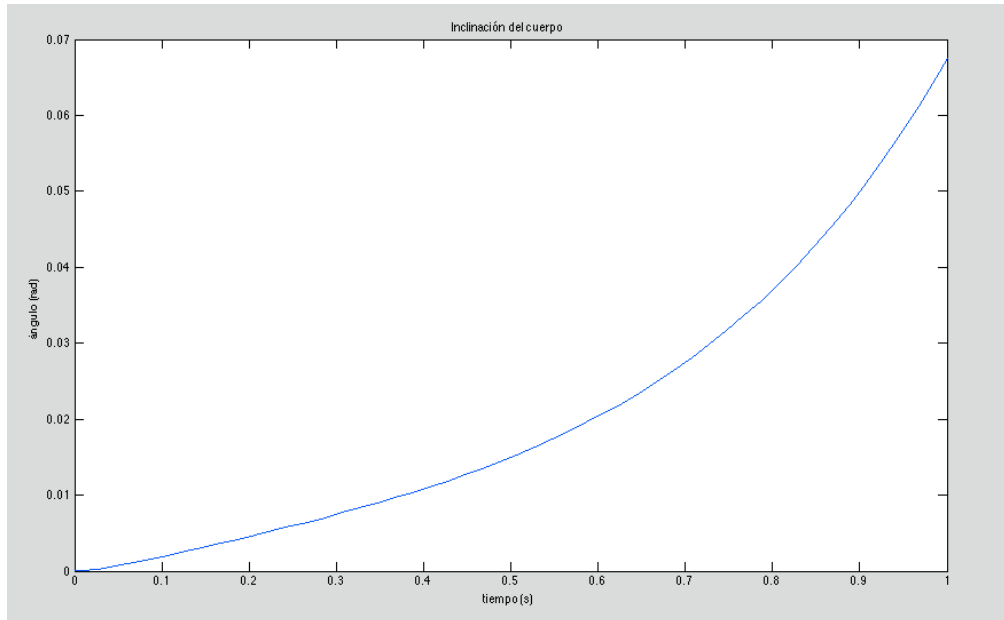


Figura 6.2 Gráfica evolución del ángulo de inclinación. Simulación en lazo cerrado.

Como se observa en la figura 6.2, la magnitud de la inclinación es menor que en el caso del sistema en lazo abierto, pero aún así el sistema sigue siendo inestable. De lo que se deduce que el sistema es demasiado complejo como para tomar en cuenta la realimentación como mecanismo de control.

### **6.1.2 Control LQR**

Se van a llevar a cabo algunas modificaciones del diseño del diagrama de bloques de cara a facilitar el análisis de los resultados de las simulaciones y que se pueden observar en la figura 6.3

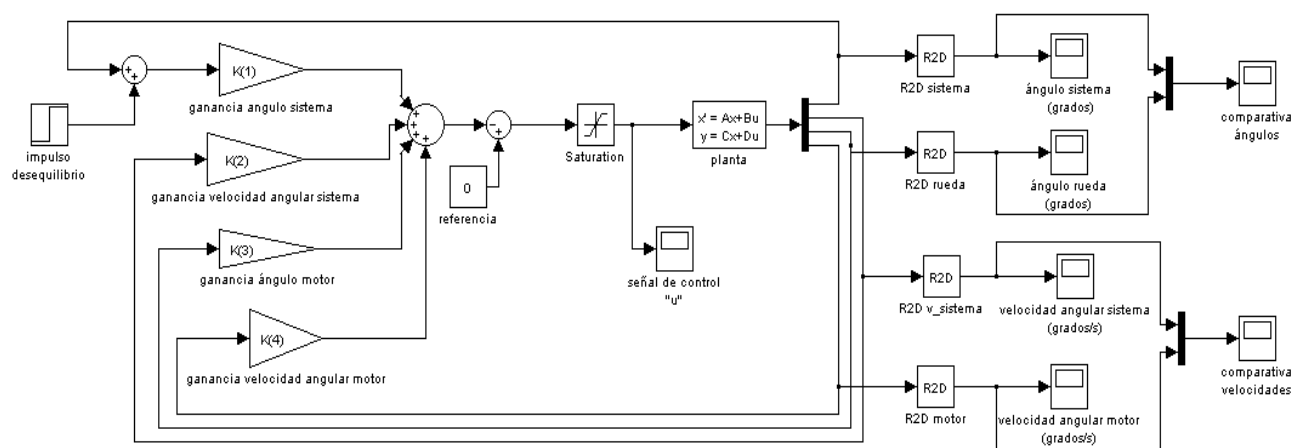


Figura 6.3 Diagrama de bloques del sistema con control LQR

En primer lugar se ha añadido un bloque de saturación. Los motores del robot se pueden programar dentro de un rango de funcionamiento, que viene dado por la potencia que se desea aplicar a las ruedas. Por tanto debemos incluir todas estas limitaciones en el modelo para lograr asemejarlo lo máximo posible al sistema real. Y como la señal de control ("u") está calculada en base a la fuerza necesaria que hay que aplicar al motor para mantener el robot en equilibrio, independientemente del valor que vaya a obtener el controlador, ésta señal siempre va a estar acotada dentro de un rango de valores. Por ejemplo, en la práctica existe un ángulo de inclinación límite a partir del cuál la estabilidad del robot es insalvable. Teóricamente sí se puede encontrar un valor que pueda alcanzar esa estabilidad, pero el servomotor del robot posee una potencia máxima aplicable, y de ahí la acotación.

Además en el diseño se han incluido conversores de radianes, medida con la que trabaja el modelo matemático, a grados para facilitar todas las lecturas. Y por último se han añadido nuevos *scopes* sobre variables que son necesarias en el análisis del resultado de la simulación como puede ser en la señal de control, o para realizar comparativas entre las salidas del sistema.

Como se ha descrito en la sección 5.1 de la memoria, se va a incluir un regulador LQR con el objetivo de controlar del robot. El regulador está compuesto por cuatro ganancias, una para cada respectiva señal retroalimentada. El procedimiento que se sigue para calcular dichas ganancias puede ser de dos formas, una de ellas sería hacer todos los cálculos necesarios que vienen especificados en el punto 5.1 de la memoria. Para simplificarlo, en



nuestro caso, vamos a hacer uso de una función de la librería de Matlab que nos calcula los valores de los “pesos” en función de nuestras matrices de estado y las matrices Q y R que especifiquemos.

Las matrices A, B, C y D son conocidas, pues son las del modelo del punto 4.1.2 de la memoria sin embargo los valores de Q y R se van a obtener del siguiente modo: para la obtención de la matriz Q, se va a partir de la matriz obtenida de la operación  $Q = C'C$  y para R se va a partir del valor  $R = 1$  que componen el caso más simple para calcular las ganancias del regulador LQR [Michigan Engineering]. En nuestro caso, la matriz Q queda como la matriz identidad y un ajuste más fino del controlador se consigue haciendo variaciones sobre los valores no nulos de dicha matriz [Michigan Engineering] mediante ensayos de prueba y error.

Y los mejores resultados se obtienen con:

$$Q = \begin{bmatrix} 50 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
$$R = 1$$

En las siguientes figuras se muestran las simulaciones realizadas en base a someter al robot a una perturbación de diez grados.

En la figura 6.4 ya podemos observar que existe un mecanismo de control que trata de mantener la estabilidad del robot. Debido a la perturbación de 10 grados el robot comienza a inclinarse pero inmediatamente el controlador trata de salvar la caída que antes era inevitable aplicando potencia en los motores eléctricos. Si ampliamos en tiempo de duración de la prueba observamos que el robot finalmente se estabiliza:

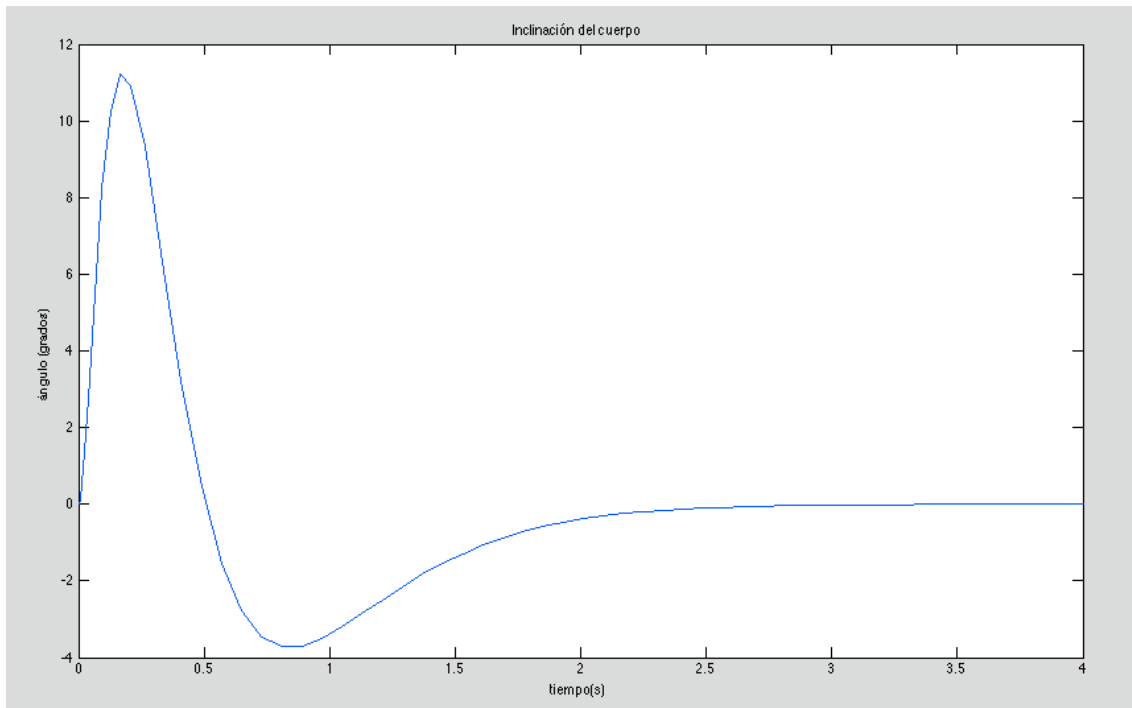


Figura 6.4 Gráfica evolución del ángulo de inclinación. Simulación control LQR.

Efectivamente, al cabo de tres segundos y medio, y después de la correspondiente sobre-elongación, debido a un exceso de potencia aplicada sobre las ruedas, el robot se estabiliza verticalmente.

Resulta interesante también observar la velocidad angular de las ruedas (Figura 6.5). Como se puede observar, inmediatamente después de comenzar a inclinarse el robot, las ruedas, por efecto de la fuerza aplicada a los servomotores, comienzan a girar en el mismo sentido en el que se está venciendo. De igual forma, al llegar la inclinación contraria, las ruedas giran en ese sentido hasta llegar a velocidad cero, pues el robot ya se encuentra estabilizado.

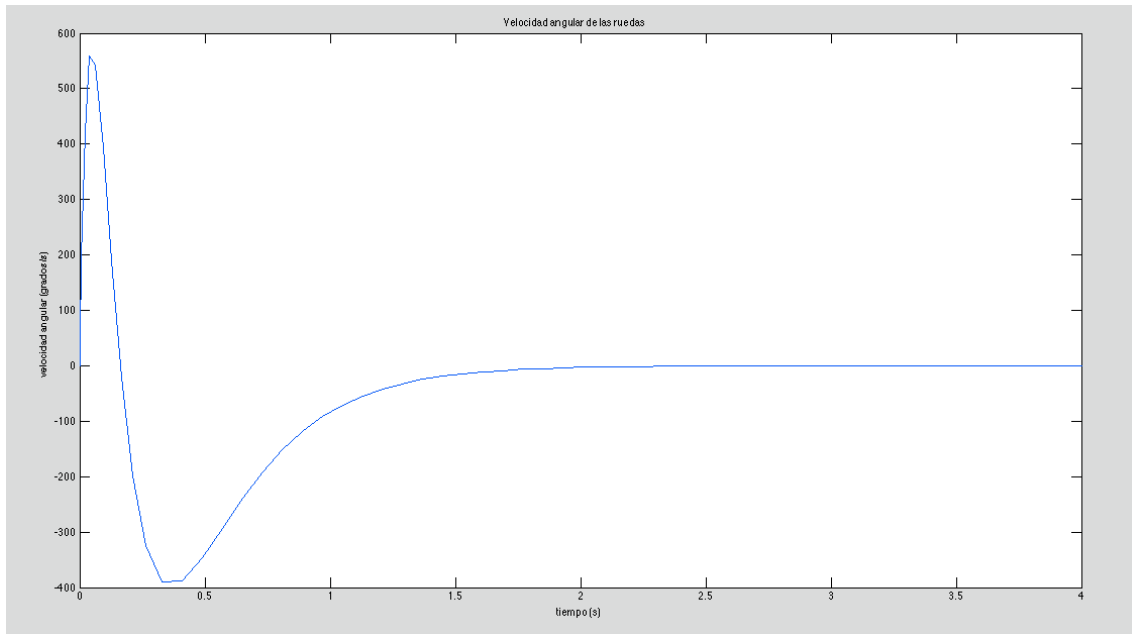


Figura 6.5 Gráfica evolución de la velocidad angular de las ruedas. Simulación control LQR.

Por tanto, ya hemos conseguido un mecanismo de control eficiente que es capaz de mantener estable nuestro robot, al menos en simulación. No obstante el siguiente paso será la conexión en serie con un controlador PID con el que pretendemos mejorar la eficiencia del control global con el objetivo de intentar que encuentre el punto de equilibrio con mayor rapidez y que produzca rebotes de menor magnitud. Podría conseguirse el mismo resultado mejorando el control que se acaba de diseñar y simular, pero el objetivo del presente proyecto no es la sintonía depurada de ninguno de ellos en especial pues conllevaría estudios más exhaustivos, sino que es la combinación de varios controladores para analizar los resultados en su conjunto.

### 6.1.3 Control LQR y PID

El objetivo es introducir un controlador PID como el que se ha descrito en la sección 5.2 entre el regulador LQR y la planta (figura 6.6) para conseguir una mejora en el control del péndulo, tanto en términos de velocidad de respuesta como en la magnitud de las sobre-elongaciones. Se pretende mejorar este aspecto ante los cambios de referencia y una disminución del tiempo de respuesta para optimizar el control tanto como sea posible.

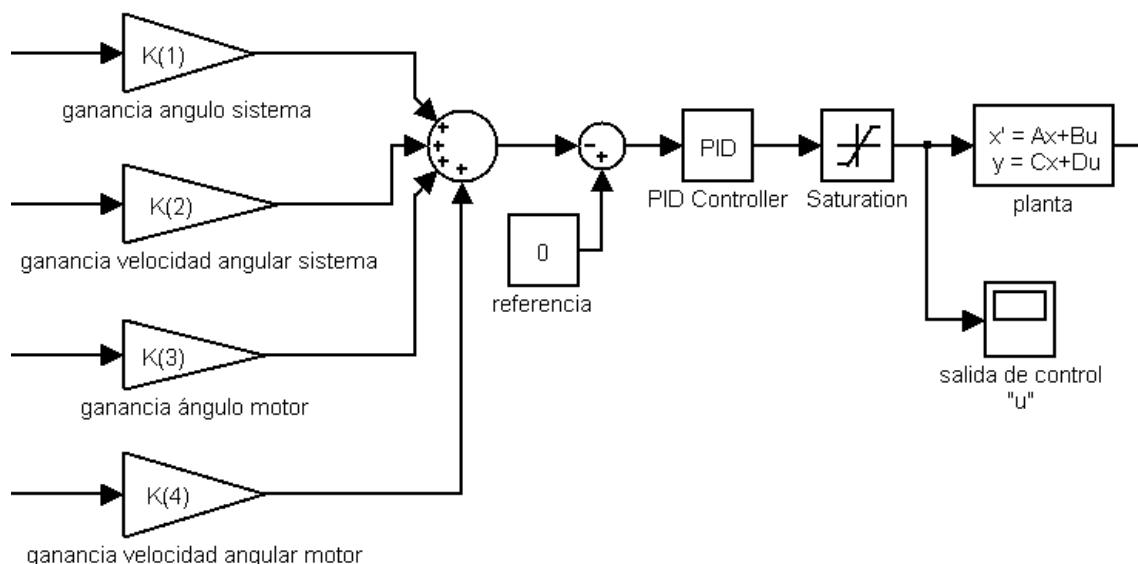


Figura 6.6 Diagrama de bloque tras la inclusión del controlador PID

En primer lugar vamos a encontrar un valor para la ganancia proporcional, que sabemos que nos va ayudar a controlar de mejor forma la sobre-elongación del sistema. Con el regulador LQR ya tenemos la retroalimentación calculada a partir de diversas constantes proporcionales, por ello sabemos que la  $k_p$  no va a adoptar un valor muy alejado de 1.

El siguiente paso es calibrar la ganancia integral  $k_i$ , teóricamente dicho valor es utilizado para eliminar el error estacionario, es decir, la diferencia entre la referencia el valor sobre el que se asienta finalmente nuestro sistema.

Por último, nos queda la constante  $k_d$ , con la que pretendemos hacer que el sistema más rápido ante cualquier cambio.

De las diferentes pruebas realizadas, los mejores resultados se obtienen con:

$$k_p=1.5$$

$$k_i=0.5$$

$$k_d=0.005$$



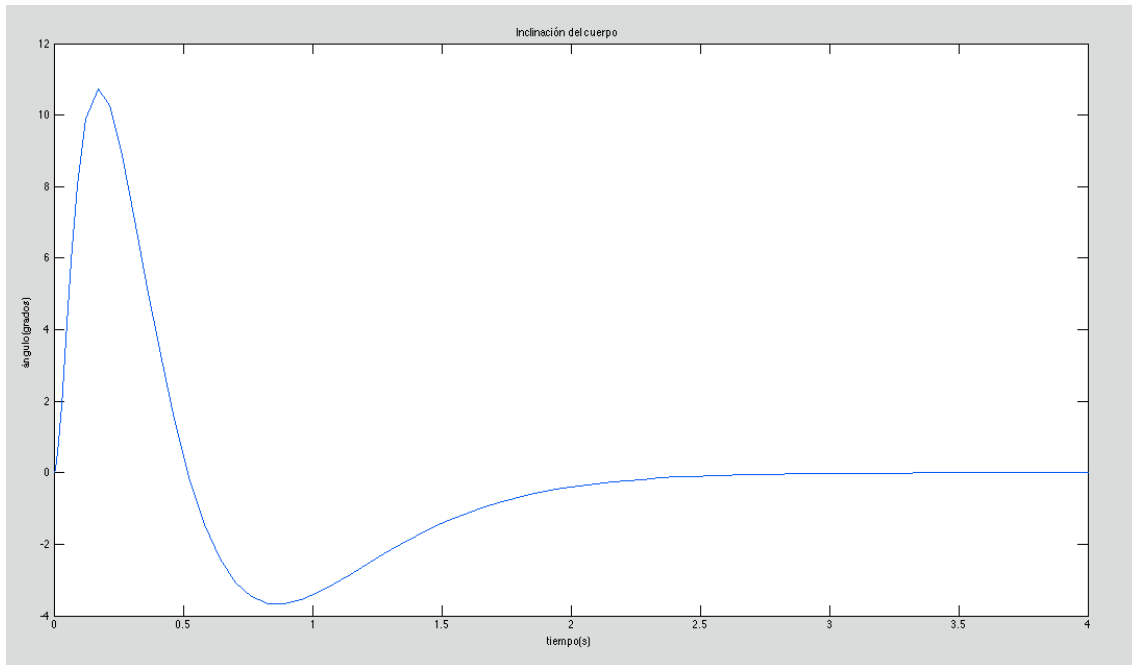


Figura 6.7 Gráfica evolución del ángulo de inclinación. Simulación control PID.

Como se observa en los resultados de la simulación, figura 6.7, con el control PID se ha conseguido disminuir la sobre-elongación, mientras que los tiempos de subida y de asentamiento apenas se han modificado.

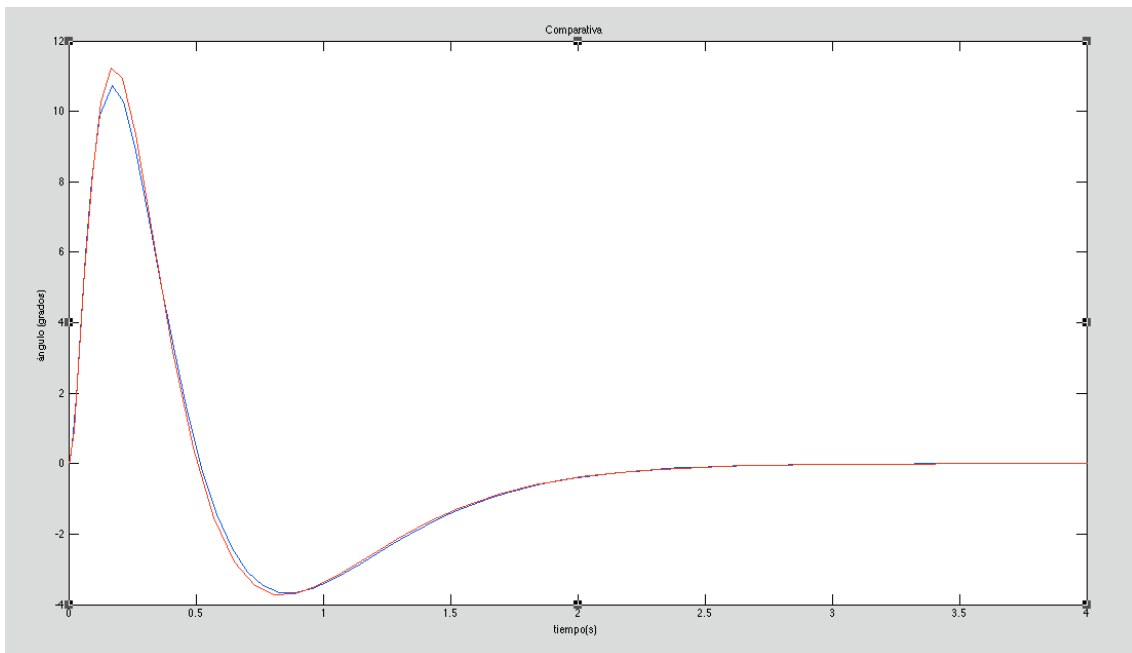


Figura 6.8 Gráfica comparativa de inclinación entre control LQR (roja) y LQR + PID (azul).



La figura 6.8 muestra una comparativa de la evolución del ángulo de inclinación del sistema con el regulador LQR y con la combinación con el controlador PID. Como se ve, ha disminuido el sobrepico, con lo que el control es más eficiente.

#### **6.1.4 Control PID de ganancia programada Fuzzy**

Como ya se ha descrito en la sección 5.2, se va a implementar un controlador PID de ganancia programada. El motivo es buscar unos valores para las ganancias del controlador que proporcionen mayor controlabilidad del robot para diferentes situaciones. Según lo descrito, se van a establecer tres zonas diferenciadas de trabajo ("A", "B" o "C") para el robot dependiendo de los valores que tomen tres señales que podemos obtener del mismo, que serán el ángulo de inclinación, la velocidad angular de inclinación y la velocidad de las ruedas que lo mueven,  $\theta_2, \dot{\theta}_2, \dot{\theta}_1$  respectivamente.

La simulación en este apartado se va a llevar a cabo en dos partes, en la primera de ellas se buscarán los valores más adecuados para las ganancias proporcional, integral y derivativa del PID para cada una de las tres zonas. Y en la segunda parte se comprobará el funcionamiento del módulo fuzzy implementado con el conjunto de reglas borrosas que se han especificado en la sección 5.3, donde también se definieron los rangos de las señales; y que nos permitirá obtener el valor de esas ganancias en función del grado de pertenencia de cada situación a cada zona de trabajo.

##### **6.1.4.1 Zonas de trabajo para el controlador PID**

Principalmente lo que va a diferenciar cada zona de trabajo a la hora de calcular la señal de control necesaria, es lo rápido y drástico que ha de ser el cambio en la potencia de los motores para intentar conseguir un control óptimo del robot. Como consecuencia, se van a analizar gráficas referentes a la evolución de la velocidad que se aplica a las ruedas.

Para lograr el objetivo, las modificaciones de las ganancias que componen el control PID se centraran principalmente en cambios en la ganancia derivativa que es la que nos va a ayudar a hacer el sistema más rápido ante los cambios, y en la ganancia proporcional para controlar excesivos sobre-picos.

## Estudio de las tres zonas de trabajo:

- Zona A

La estabilidad del robot no se ve especialmente comprometida pues la inclinación del cuerpo es baja y las velocidades de dicha inclinación y de las ruedas presenta valores mínimos. Para esta zona se va a usar los valores de las ganancias obtenidos en el punto anterior, donde se sintonizó un PID simple.

Observamos la evolución de la velocidad angular de las ruedas para esa zona de trabajo en la figura 6.9. Como se aprecia, la velocidad crece rápidamente para estabilizar el robot y a continuación acelera en sentido contrario para salvar el sobrepico en el ángulo de inclinación. Es importante señalar que las gráficas corresponden con simulaciones realizadas de 0 s a 0.5 s para apreciar con mayor claridad los cambios en las tres zonas de trabajo. La simulación está realizada con una perturbación de tres grados en la medida de giróscopo que nos da la inclinación del cuerpo del robot y se utilizará la misma en la simulación de las otras tres zonas. Teóricamente una perturbación así pertenecería a la zona de trabajo A, y no a las otras dos, pero de esta forma podemos comparar los resultados; si se partiese de condiciones diferentes no se podrían evaluar las diferencias de los resultados obtenidos.

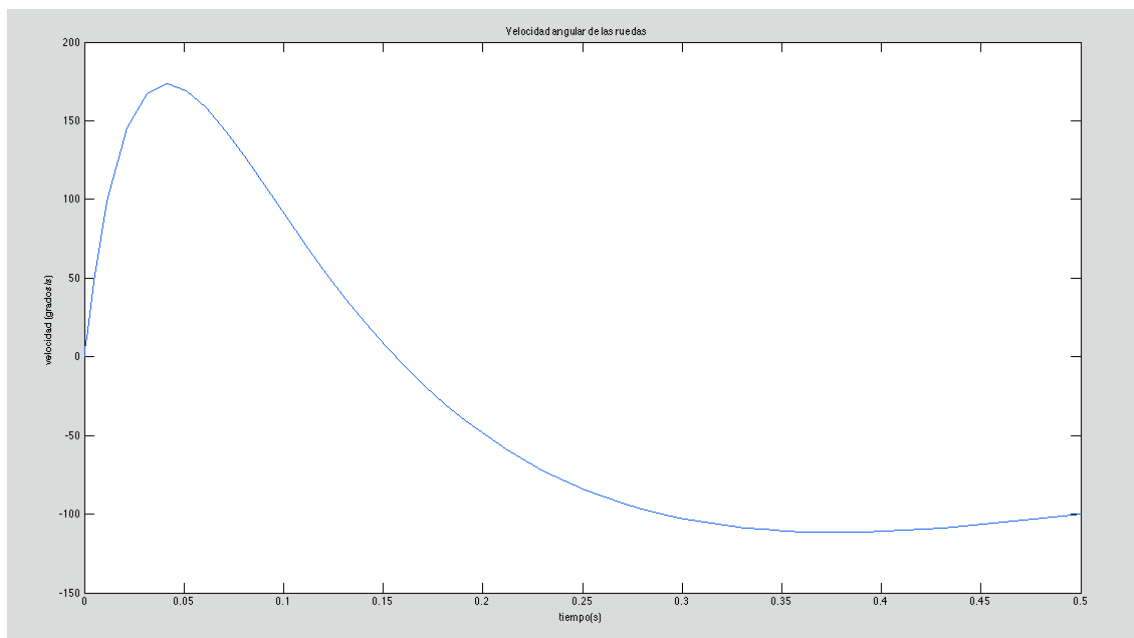


Figura 6.9 Gráfica evolución de la velocidad angular de las ruedas. Zona A.

Esta gráfica es con la que realizaremos las comparativas en las siguientes dos zonas, en las que perseguiremos que la reacción de los motores se produzca cada vez más rápido y por tanto que la aceleración

de los motores sea mayor. Más concretamente lo que buscamos en acortar el tiempo en que se produce la respuesta de los motores, como se aprecia en la figura 6.10.

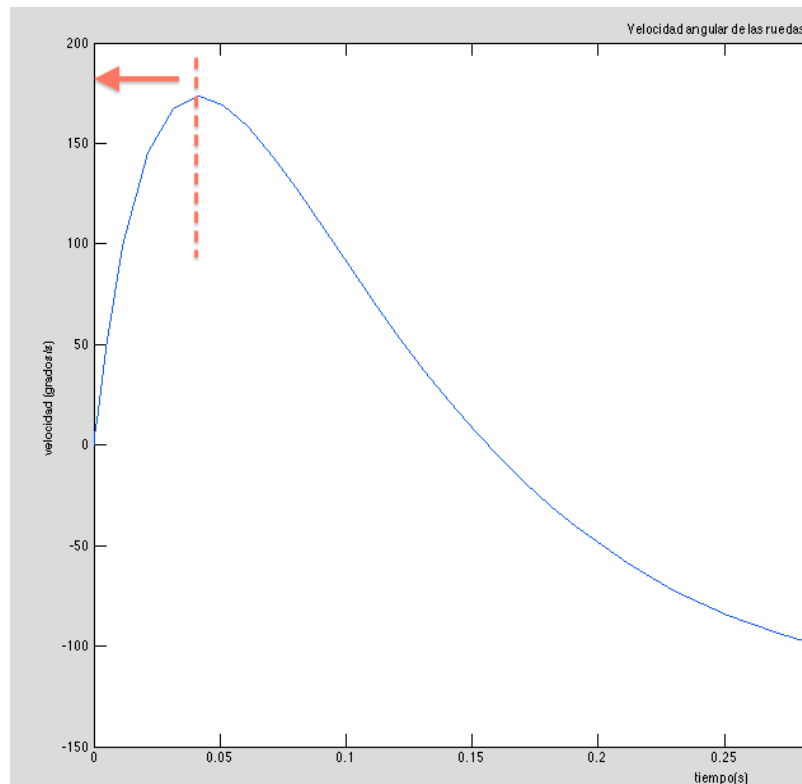


Figura 6.10 Tiempo de subida

- Zona B

Se trata de una situación más desfavorable que la anterior pues el ángulo de inclinación que presenta el cuerpo es mayor, la velocidad de caída del mismo puede que sea más elevada de lo que se presupondría estar en la zona de trabajo "A" o que la velocidad con la que se mueve el robot a través de las ruedas de por sí o en combinación con las características anteriores comprometa seriamente la estabilidad.

Se necesita pues, un control más directo y drástico. Por tanto se busca obtener gráficas en las que veamos que se reacciona con mayor rapidez pero sin conseguir reacciones extremadamente enérgicas pues podemos con ello contribuir a la desestabilización intentando estabilizar la el robot.

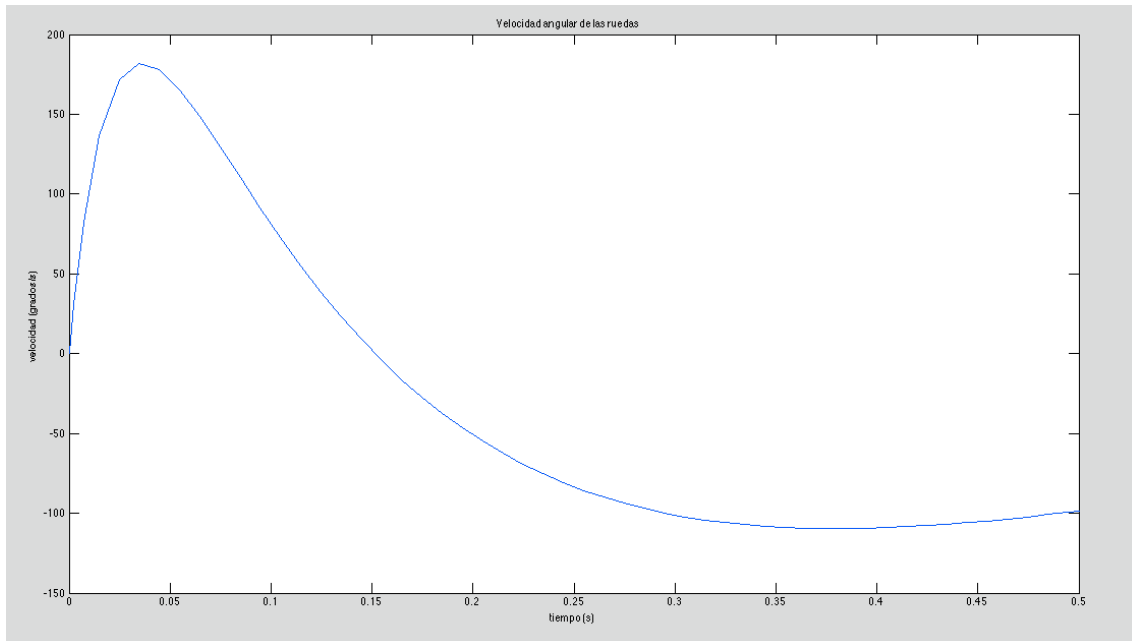


Figura 6.11 Gráfica evolución de la velocidad angular de las ruedas. Zona B.

Como se puede apreciar en la gráfica 6.11, el valor máximo que alcanza la velocidad se produce antes que en la zona "A", lo que implica que la aceleración ha aumentado por lo que mejoramos la respuesta temporal del controlador.

- Zona C

Es la zona más desfavorable en la que se puede encontrar el robot y se puede entrar en ella por variaciones agudas de las señales que se contemplan, como puede ser por ejemplo una modificación repentina y alta del ladeo del robot o porque el éste se está desplazando a alta velocidad lo que puede producir inestabilidad con facilidad.

Se buscará en este punto una respuesta lo más rápida posible ante éstos acontecimientos pues el tiempo de reacción será mínimo. De nuevo, estudiando principalmente las variaciones que se producen modificando la ganancia derivativa del PID obtenemos el siguiente resultado (figura 6.12):

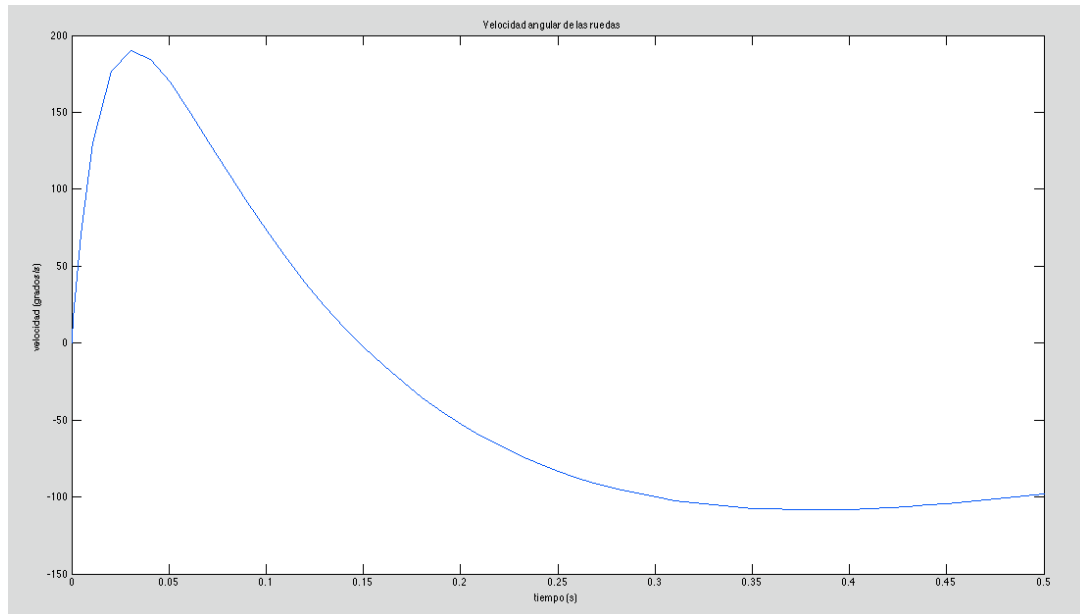


Figura 6.12 Gráfica evolución de la velocidad angular de las ruedas. Zona C.

La reacción de los motores se produce con mayor rapidez que en la zona de trabajo "B", logrando el aumento de velocidad repentino que se persigue para salvar la situación.

Podemos comparar los resultados con mayor claridad en la figura 6.13, observando que cada zona de trabajo en que se sitúe el robot proporciona una aceleración de los motores diferente.

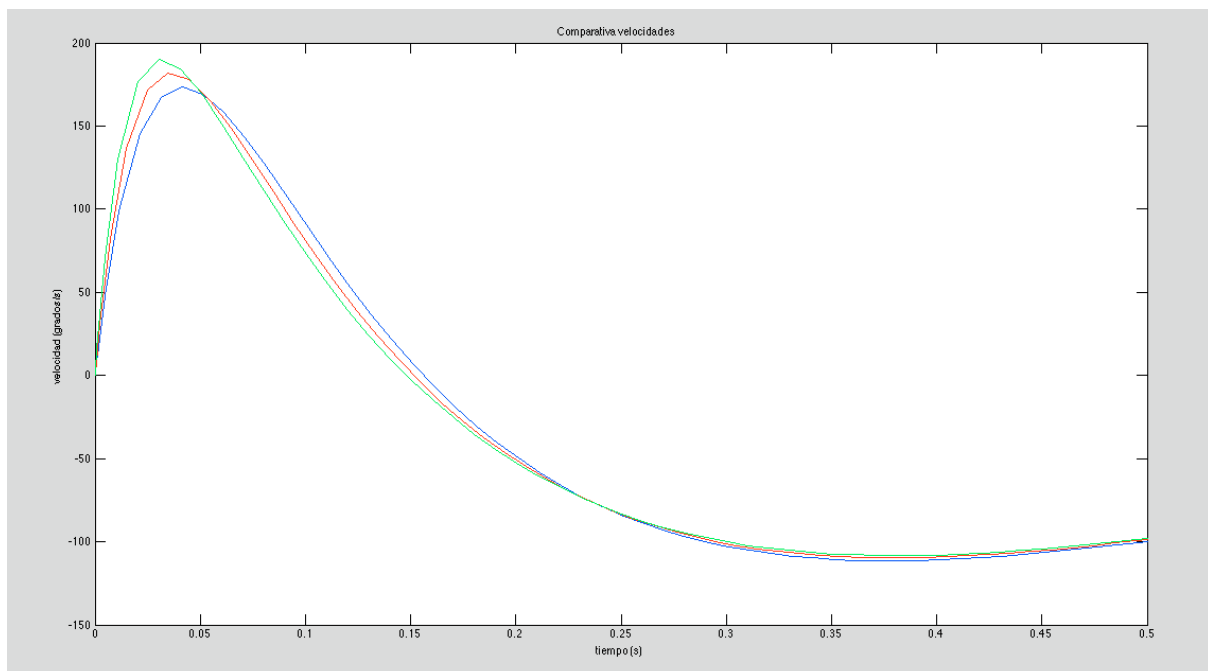


Figura 6.13 Gráfica comparativa de las velocidades angulares aplicadas a las ruedas para las tres zonas

La gráfica azul se corresponde con la zona A, la roja con la zona B y la verde representa la velocidad en la zona C.

Ya tenemos por tanto los valores que buscábamos para las ganancias que componen un PID para cada área de trabajo donde se puede situar el robot, resultando las siguientes:

	Zona A	Zona B	Zona C
Kp	1.5	1.75	2
Ki	0.5	0.5	0.5
Kd	0.005	0.003	0.001

Tabla 6.1 Ganancias para cada área

#### 6.1.4.2 Módulo Fuzzy

Para comprobar el funcionamiento del módulo fuzzy diseñado en la sección 5.3, el proceso pasa por convertir en diagrama de bloques dentro de la herramienta Simulink de Matlab el archivo “.fis”; donde se encuentran los rangos que tienen las señales de entrada y de salida, así como el conjunto de reglas que permita conocer el grado de pertenencia de los valores de entrada para cada área de trabajo (Figura 6.14).

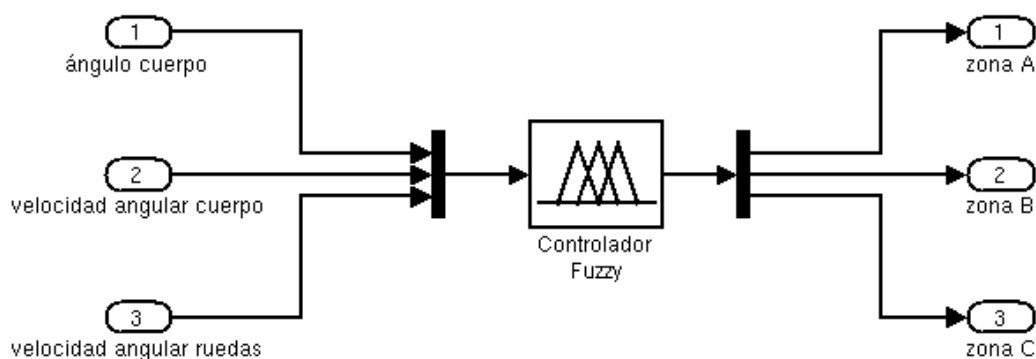


Figura 6.14 Esquema de entradas y salidas del módulo fuzzy.

De este modo ya podemos averiguar para un determinado valor de las variables de entrada su valor proporcional dentro de cada una de las tres zonas de trabajo. Una vez que conocemos esa relación de pertenencia sólo nos queda calcular el valor total que adopta cada ganancia del controlador PID. El cálculo de cada ganancia viene dado por el sumatorio del valor que tiene esa ganancia en esa zona de trabajo multiplicado por la pertenencia a esa zona:

$$k_p = Ak_{pA} + Bk_{pB} + Ck_{pC}$$

$$k_i = Ak_{iA} + Bk_{iB} + Ck_{iC}$$

$$k_d = Ak_{dA} + Bk_{dB} + Ck_{dC}$$

Cálculos que a su vez podemos implementar mediante simples diagramas de bloques, tanto para Kp como para Kd ya que como se ha visto, Ki no varía sea el área que sea:

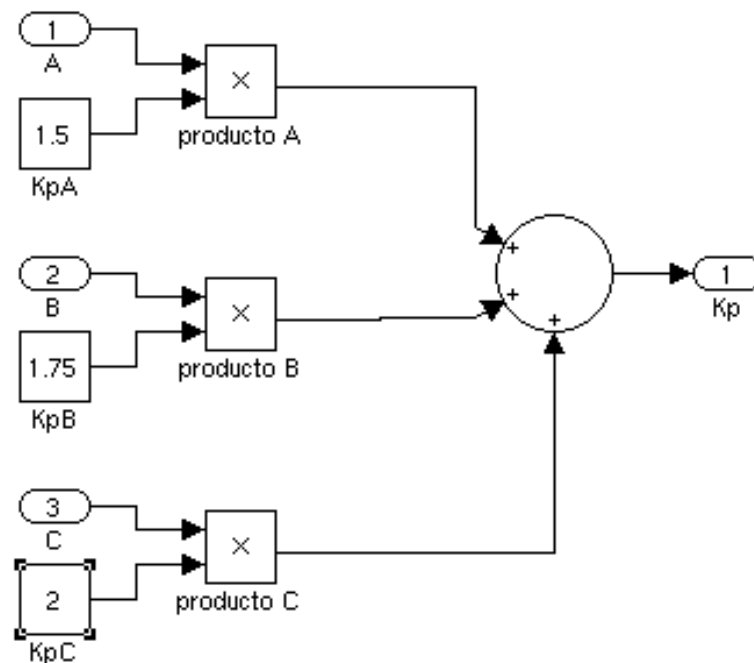


Figura 6.15 Esquema de cálculo de la ganancia Kp del módulo PID



Esquema correspondiente al cálculo de  $k_p$  (Figura 6.15), y del mismo modo para  $k_d$  (Figura 6.16):

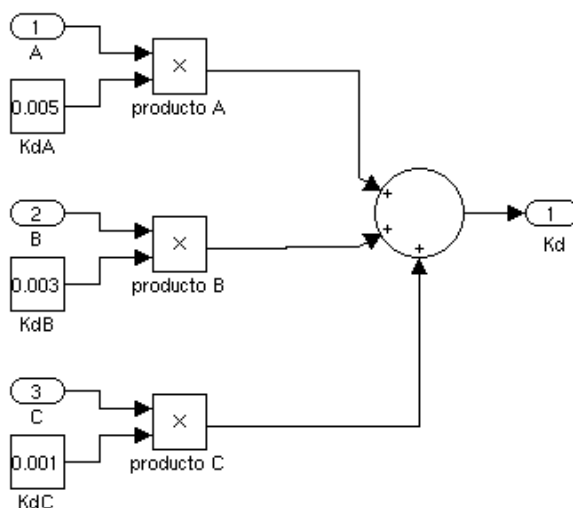


Figura 6.16 Esquema de cálculo de la ganancia  $K_d$  del módulo PID

Se puede observar como evolucionan las ganancias proporcional, figura 6.17, y derivativa ante diferentes situaciones dependiendo del grado de pertenencia a cada zona.

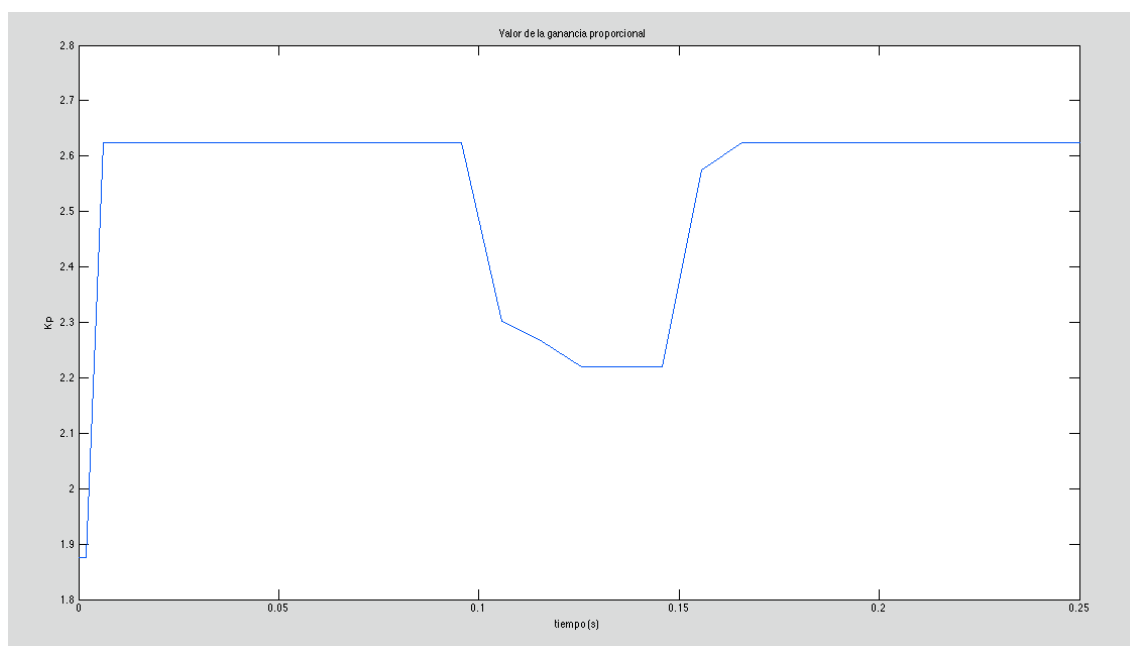


Figura 6.17 Evolución de la ganancia  $K_p$  al pasar por diferentes zonas de trabajo.

La figura 6.18 ilustra los valores que adopta  $k_d$ , la ganancia derivativa, como se puede observar llega a valores cercanos a 0.005 pero también baja hasta prácticamente 0.002, lo que ejemplifica el paso del robot por varias de las zonas de trabajo, en este caso, los grados de pertenencia de las zonas A y B son las más representativas.

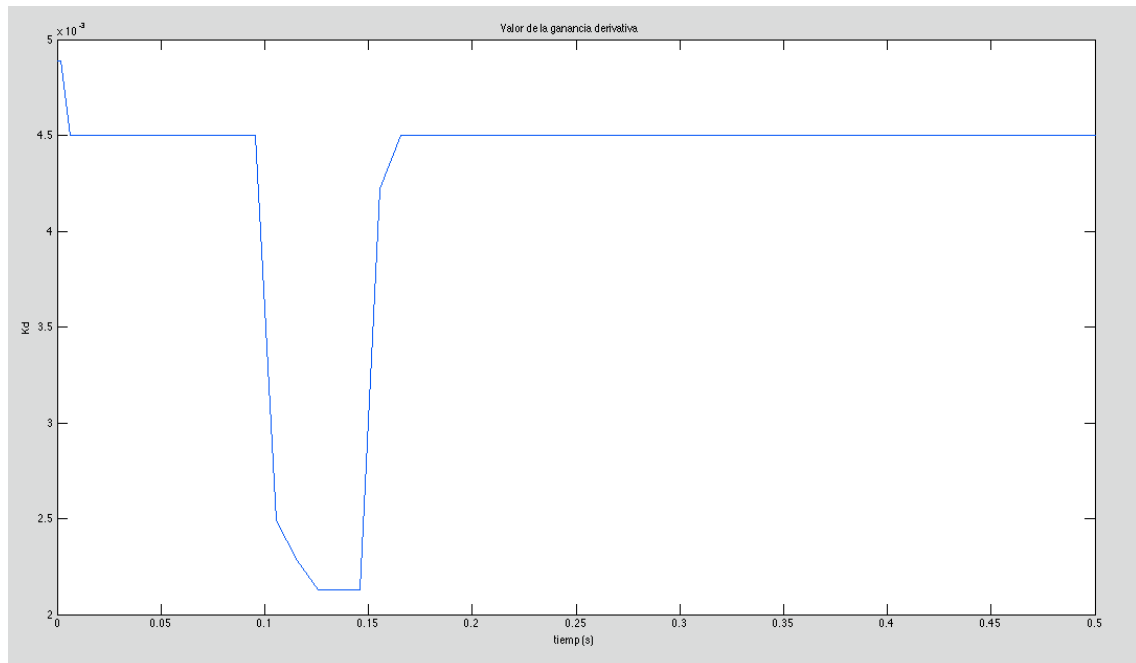


Figura 6.18 Evolución de la ganancia  $K_d$  al pasar por diferentes zonas de trabajo.

## 6.2 Implementación del control en el sistema real

Para la implementación de los controladores se ha elegido la plataforma RobotC<sup>10</sup> la cual dispone de librerías para trabajar con el procesador NXT. Este entorno se basa en el lenguaje de programación C y ofrece una serie de funciones ya implementadas al usuario para la lectura de los diferentes sensores que pueden ser conectado al procesador NXT, también dispone de una interfaz de recogida de datos en tiempo real de los sensores incorporados al robot mientras éste está en funcionamiento. Además ofrece la opción de

<sup>10</sup> **RobotC:** entorno de desarrollo integrado dirigido a los estudiantes que se utiliza para programar y controlar LEGO, VEX y los robots RCX que utilizan un lenguaje de programación basado en el lenguaje de programación C



controlar el robot mediante bluetooth<sup>11</sup>, tecnología inalámbrica para la cual está ya preparado el procesador NXT. Además esta plataforma cuenta con una gran comunidad de usuarios y desarrolladores, con un gran número de foros en los que poder apoyarse en caso de surgir complicaciones con la implementación de los controladores, por todo ello esta plataforma fue la elegida frente a otras posibles opciones como LeJOS NXJ<sup>12</sup> o NXT\_Python<sup>13</sup>

Una vez diseñados y sintonizados los controladores para el robot, pasamos a traducir todas sus acciones a código compilable para el procesador NXT, partiremos de la estructura básica del PID, calculando el error de sistema para la componente proporcional de control, y derivándolo e integrándolo para la componentes derivativa e integral respectivamente.

En RobotC es un lenguaje orientado a tareas, se codifican diferentes tareas que serán compiladas y cargadas en el procesador del robot para ser ejecutadas concurrentemente.

En nuestro caso, para garantizar la estabilidad del robot hemos codificado una única tarea encargada de mantener el equilibrio del cuerpo del robot, adicionalmente se ha codificado una función para el calibrado del giroscopio del robot antes de trabajar con él.

Para realizar los cálculos en el código se han definido una serie de constantes en el código, por ejemplo para los parámetros de las ganancias del LQR para cada variable del sistema, que codifican la importancia de cada variable en el sistema. Se definen también constantes para los valores de las ganancias proporcional, derivativa e integral para cada una de las tres zonas de trabajo, y para el diferencial de tiempo con el que trabajamos, también los valores de velocidad y aceleración con los que queremos que trabajen los motores de las ruedas del robot y el diferencial de tiempo, con todo ello la declaración de las constantes más relevantes del código de control quedan mostradas en la siguiente tabla 6.2:

---

<sup>11</sup> **Bluetooth:** es una especificación industrial para Redes Inalámbricas de Área Personal (WPAN) que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda ISM de los 2.4 GHz

<sup>12</sup> **LeJOS NXT:** Incluye una máquina virtual Java, que permite a los robots de Lego Mindstorms a programar en el lenguaje de programación Java.

<sup>13</sup> **Python:** es un lenguaje de programación de alto nivel cuya filosofía hace hincapié en una sintaxis muy limpia y favorezca un código legible. Se trata de un lenguaje de programación multiparadigma ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación imperativa.



<b>Descripción</b>	<b>Constante</b>	<b>Valor</b>
Ganancia LQR Ángulo Robot	gn_th	122.9601
Ganancia LQR Velocidad Balanceo Robot	gn_dth_dt	36.6719
Ganancia LQR Ángulo Rueda	gn_y	-7.0711
Ganancia LQR Velocidad de la Rueda	gn_dy_dt	-5.4398
Diferencial de Tiempo	dt	0.010s
Aceleración (estático)	Aceleración	0
Velocidad (estático)	Velocidad	0
Ganancia Proporcional Zona A PID	KpA	1.5
Ganancia Derivativa Zona A PID	KdA	0.005
Ganancia Integral Zona A PID	KiA	0.5
Ganancia Proporcional Zona B PID	KpB	1.75
Ganancia Derivativa Zona B PID	KdB	0.003
Ganancia Integral Zona B PID	KiB	0.5
Ganancia Proporcional Zona C PID	KpC	2
Ganancia Derivativa Zona C PID	KdC	0.001
Ganancia Integral Zona C PID	KiC	0.5

Tabla 6.2 Constantes

Como variables más importantes del sistema tenemos las cuatro variables de control, el ángulo del cuerpo, la velocidad angular de balanceo del cuerpo del robot, la velocidad de las ruedas y las posiciones de las ruedas. Otra variable importante es la referencia de medida en reposo del giroscopio, la cual utilizaremos para evitar errores acumulados de medida. También es una variable de importancia la que usaremos para transmitir potencia a los motores. con todo ello las variables más relevantes del sistema quedan dispuestas en la siguiente tabla 6.3:



Descripción	Variable
Ángulo del cuerpo del robot	th
Velocidad de balanceo del robot	dth_dt
Posición de la rueda	y
Velocidad de la rueda	dy_dt
Error	e
Derivada del error	de_dt
Integral del error	_edt
Medida en reposo del giroscopio	referencia
Señal de control	u

Tabla 6.3 Tabla de variables

La función encargada de calibrar el robot realizará 40 lecturas acumuladas del giroscopio instalado en el robot, el cual deberá estar en reposo sobre una superficie sin vibraciones ni movimientos, entre cada par de lecturas se esperan 50 milisegundos, al finalizar las lecturas se dividirá el valor acumulado por 40 para hallar el valor promedio que mide el giroscopio en reposo. Desde el código principal se llama a esta función y se recoge su resultado sobre la variable "referencia". Estos cálculos pueden observarse en el siguiente fragmento de código (Listado 6.1).



```
//Funcion para el calibrado del giroscopio
int calibrar_giroscopio(){

    //Funcion para calibrar el giroscopio con el offset en reposo
    int lectura_giroscopio = 0, i = 0;

    while(nNxtButtonPressed == kEnterButton){}
    nxtDisplayTextLine(0,"Coloque el robot");
    nxtDisplayTextLine(1,"en reposo");
    wait1Msec(500);
    nxtDisplayTextLine(2,"Calibrando");
    nxtDisplayTextLine(3,"giroscopio");

    for(i = 0; i < 40;i++){
        lectura_giroscopio = lectura_giroscopio + SensorRaw[Gyro];
        wait1Msec(50);}
    lectura_giroscopio = lectura_giroscopio/40;
    PlayTone(500,50);

    if(lectura_giroscopio < 550 || lectura_giroscopio > 640){
        nxtDisplayTextLine(4,"Error");
        nxtDisplayTextLine(6,"Sensor conectado?");
        wait1Msec(2000);StopAllTasks();}

    eraseDisplay();
    nxtDisplayTextLine(2,"Giroscopio");
    nxtDisplayTextLine(4,"calibrado");
    while(nNxtButtonPressed != kEnterButton){}
    while(nNxtButtonPressed == kEnterButton){}
    eraseDisplay();

    return(lectura_giroscopio);
}
```

Listado 6.1 Función de calibrado

La tarea principal de control la llamaremos "equilibrio", en ella codificamos la estructura del controlador PID, en la variable del error acumulamos las cuatro variables del sistema, multiplicadas cada una por su ganancia LQR, posteriormente derivamos e integramos esta variable de error para hallar las componentes derivativa e integral respectivamente. Con estas variables y sus ganancias calculadas por la lógica borrosa se calcula una suma, que será la señal de potencia que se enviará a los motores del robot.

Para el cálculo de la velocidad angular del balanceo del cuerpo del robot, variable `dth_dt`, realizamos dos lecturas del giroscopio y calculamos la media. Una vez calculada esta variable, podemos calcular el ángulo del cuerpo del robot sumándole al valor anterior (inicialmente 0 al comenzar el programa) la velocidad angular calculada multiplicada por el diferencial de tiempo. Una vez realizados estos cálculos se actualiza el valor de la referencia del giroscopio. El siguiente fragmento de código (Listado 6.2) recoge los cálculos descritos.

```
//Calculo del angulo y actualizacion de la referencia
dth_dt = datos_giro/2 - referencia;
referencia = referencia*0.999 + (0.001*(dth_dt+referencia));
th = th + dth_dt*dt;
```

Listado 6.2 Cálculo del ángulo del cuerpo del robot

El siguiente paso es calcular las variables de velocidad y posición de las ruedas del robot. Para ello calculamos la posición referencia que deberá tener la rueda del robot en base a la velocidad y aceleración con las que hemos definido que debe trabajar el robot. Esta posición de referencia se traduce a los valores manejados por los encoders de los motores, y teniendo en cuenta la variación de posición de estos encoders y el diferencial de tiempo, calculamos la velocidad de movimiento real de las ruedas. Estos cálculos se observan en el siguiente fragmento de código (Listado 6.3).

```
//Control de la vel, acel y desplazamiento del movimiento del robot
if(v < velocidad*10){
v = v + aceleracion*10*dt;}
else if(v > velocidad*10){
v = v - aceleracion*10*dt;}

y_ref = y_ref + v*dt;

//Control de la posicion del motor y de la rueda
n++;if(n == n_max){n = 0;}
encoder[n] = nMotorEncoder[motorB] + nMotorEncoder[motorC] + y_ref;
n_comp = n+1;if(n_comp == n_max){n_comp = 0;}
y = encoder[n]*degtorad*radio/rel_motor_rueda;
dy_dt = (encoder[n] - encoder[n_comp])/(dt*(n_max-1))*degtorad*radio/rel_motor_rueda;
```

Listado 6.3 Cálculo de la posición y velocidad de la rueda

Una vez calculados los valores para las cuatro variables del sistema pasamos a calcular el valor del error, para ello realizamos la suma de las

variables, cada una de ellas multiplicada por su ganancia LQR. Una vez calculado el error, podemos calcular su derivada, calculando su variación respecto de su valor anterior y el diferencial de tiempo, y su integral, acumulando su valor en una variable, estos cálculos son los utilizados para obtener los valores de las componentes proporcional, derivativa e integral del controlador PID, y se pueden observar en el siguiente fragmento de código (Listado 6.4).

```
//Calculo de las componentes proporcional, derivativa e integral del PID
e = gn_th * th + gn_dth_dt * dth_dt + gn_y * y + gn_dy_dt * dy_dt;
de_dt = (e - e_prev)/dt;      // Calculamos la derivada de error
_edt = _edt + e*dt;          // Calculamos la integral de error
e_prev = e;
```

Listado 6.4 Cálculo del error, derivada e integral

Pasamos ahora a calcular la señal de control del sistema, para ellos multiplicamos cada componente del PID por su ganancia correspondiente, que dependerá de la señal ofrecida por la lógica borrosa, este cálculo puede observarse en el siguiente fragmento de código (Listado 6.5).

```
//Calculo de la señal de control

u = ((kpA*PA+kpB*PB*kpC*PC) *e + (kiA*PA+kiB*PB*kiC*PC) *_edt +
      (kdA*PA+kdB*PB*kdC*PC) *de_dt)/radio*rel_motor_rueda;
```

Listado 6.5 Cálculo de la señal de control

En función del sentido del movimiento, y para poder realizar los giros se realiza el cálculo de la relación de potencia transmitida a cada uno de los motores en función de la distancia que deba recorrer cada rueda en el giro y se transmite dicha potencia para mover el robot, esto puede observarse en el siguiente fragmento de código (Listado 6.6).





```
//Control del sentido del movimiento
if(sentido == 0){
    if(prev_sentido != 0){
        straight = nMotorEncoder[motorC] - nMotorEncoder[motorB];
        d_pwr = (nMotorEncoder[motorC] - nMotorEncoder[motorB] - straight)/(radio*10/rel_motor_rueda);
    }else{d_pwr = sentido/(radio*10/rel_motor_rueda);}
    prev_sentido = sentido;

//Control de la potencia suministrada a los dos motores
motorpower = u;
motor[motorB] = motorpower + d_pwr;
motor[motorC] = motorpower - d_pwr;
```

Listado 6.6 Cálculo de la potencia aplicada a los motores

Finalmente, para acotar la zona de trabajo del controlador, definimos unos valores de saturación del sistema, dentro de los cuales será donde trabaje el controlador, pero que si son superados, se abortará la acción de control, como puede observarse en el siguiente fragmento (Listado 6.7).

```
//Acotamiento de la zona de trabajo del controlador PID
if(abs(th)>30 || abs(motorpower) > 2000){
    StopAllTasks();
}
```

Listado 6.7 Saturación del controlador

Con todo esto queda implementado en el robot un controlador PID de ganancia programada que trabaja en acción conjunta con un LQR que regula los pesos de las variables del sistema. Las ganancias del PID las define una lógica borrosa en función de las variables de sistema.



## 7 Conclusiones y trabajo futuro

### 7.1 Conclusiones

Después de un año trabajando en el proyecto y con la experiencia que se ha adquirido, creemos que es básico la obtención de un modelo matemático del sistema físico que cumpla cierto compromiso entre complejidad y exactitud para poder simularlo para sintonizar las estructuras de control, ya que para la obtención de nuestro modelo hemos decidido obviar detalles como el nivel de la batería que influye en el comportamiento del robot, pero que hace mucho más compleja la obtención del modelo en relación a la precisión que añade.

Se ha cumplido el objetivo de diseñar un controlador para el péndulo que combina distintas estrategias de control para garantizar el equilibrio, concretamente pretendíamos implementar un PID de ganancia programada, cuyas zonas de trabajo sean seleccionadas mediante lógica borrosa, el cual trabaja conjuntamente con un regulador LQR.

Hemos tenido algunos problemas en el desarrollo del proyecto, como por ejemplo la estimación de la inclinación del ángulo de robot mediante la lectura del giroscopio ya que este sensor mide velocidades angulares y con cierto ruido.



Figura 7.1 Prototipo en la posición inicial



Figura 7.2 Prototipo avanzando



Figura 7.3 Prototipo subiendo una pendiente

## **7.2 Trabajos futuros**

Como posibles ampliaciones del proyecto, en un futuro se podrían integrar otros componentes para poder mejorar el modelo como los sensores de inclinación y de vibración de Lego que son cómodos y sencillos de conectar, para mejorar la maniobrabilidad del robot en superficies más irregulares.

Por otro lado, el robot se podría mejorar su funcionamiento utilizando sensores de proximidad o ultrasonidos con el fin de seguir trayectorias dentro de un circuito y detectar y evitar obstáculos con los que pueda chocar, así como transmisores de radio o bluetooth para poder controlarlo a distancia.

Ampliando la configuración con un sensor de presión podría capacitarse al robot para trabajar en entornos sometidos a variaciones bruscas en la presión, una posible aplicación de esto podría estar en el uso del robot en el mantenimiento de maquinaria industrial en tanques de altas presiones, que en combinación con el uso de sensores infrarrojos y sensores de visión, serviría para el diagnóstico de averías en entornos hostiles al ser humano, o de difícil acceso, reduciendo el riesgo de parada en las cadenas de producción.





## 8 Referencias bibliográficas

[Anderson, 2003]

D. P. Anderson "Nbot Balancing Robot", 2003. Disponible en <http://geology.heroy.smu.edu/~dpa-www/robo/nbot>.

[Furuta, 1976]

K. Furuta, H. Nishihara and S. Mori. "Control of Unstable Mechanical Systems: Control of Pendulum". International Journal of Control, Vol. 23, pp. 673-692, 1976.

[Grasser, 2002]

F. Grasser, A. D'Arrigo, S. Colombi and A. Rufer. "Joe: A Mobile, Inverted Pendulum". Swiss Federal Institute of Technology, 2002.  
[http://leiwwww.epfl.ch/publications/grasser\\_darrigo\\_colombi\\_rufer\\_mic\\_01.pdf](http://leiwwww.epfl.ch/publications/grasser_darrigo_colombi_rufer_mic_01.pdf)

[Hassenplug, 2002]

S. Hassenplug. "Legway", 2002. Disponible en <http://www.teamhassenplug.org/robots/legway>.

[Hurbains, 2007]

P. Hurbain's. "Nxtway", 2007. <http://www.philohome.com/nxtway/>.

[Kosko, 1992]

Kosko B. Neural Networks and Fuzzy Systems: A Dynamical System S Approach to Machine Intelligence Eaglewood. Prentice hall 1992

[Kuo, 1992]

Kuo, Benjamin. Digital Control Systems. Second edition. Oxford University Press. New York. 1992

[Michigan Engineering]

<http://www.engin.umich.edu/group/ctm/examples/pend/invSS.html> Michigan Engineering

[Novakowski, 2006]

N. Novakowski, "Equations os motion and control of an inverted pendulum", Rensselaer al Harford, CT, Tech. Rep. 2006.



[Ogata, 1997]

Ogata, Katsuhiko. Modern Control Engineering. 3<sup>o</sup> Edition. Prentice Hall. 1997

[Ooi, 2003]

R. Ooi. "Balancing a Two-Wheeled Autonomous Robot". University of Western Australia, 2003. Disponible en <http://robotics.ee.uwa.edu.au/theses/2003-Balance-Ooi.pdf>

[Sherman, 2003]

B. Sherman. "Balibot, an Inverted Pendulum Robot", 2003. Disponible en <http://home.comcast.net/~botronics/balibot.html>.

[Stimac, 1999]

A. K. Stimac. (1999). Standup and stabilization of the inverted pendulum, B.S. thesis, Dept. Metch. Eng., Massachusetts Inst.Technol. Cambridge, MA.

[Watanabe, 2007]

R. Watanabe. "Motion Control of Nxtway Lego Segway". Universidad de Waseda, Japón, 2007. [http://web.mac.com/ryo\\_watanabe/](http://web.mac.com/ryo_watanabe/)

[Zadeh, 1965]

Zadeh L.A Fuzzy Sets. Information Control 1965